Special Issue on Iterative Methods in Numerical Linear Algebra

This issue of *SISC* is dedicated to papers that were presented at the Colorado Conference on Iterative Methods, which was held in Breckenridge, Colorado, April 5–9, 1994. This was the ninth in the series of Copper Mountain Conferences, the third dedicated to general iterative methods, and the first held outside Copper Mountain. Although the two locations are less than 20 miles apart, Breckenridge offered a new setting complete with an historic mining town that was established over a century ago.

The meeting was attended by approximately 180 mathematicians from all over the world who presented 115 talks on a broad range of subjects. The talks were organized into morning and afternoon sessions leaving the afternoons free for informal discussions. Loosely grouped by topic, the sessions included

Domain Decomposition (8 talks)
Nonlinear Problems (7 talks)
Integral Equations and Inverse Problems (4 talks)
Eigenvalue Problems (3 talks)
Nonsymmetric System Solvers (16 talks)
Parallel Computation (16 talks)
Iterative Method Theory (4 talks)
Software and Programming Environments (4 talks)
ODE Solvers (4 talks)
Student Papers (3 talks)
Multigrid and Multilevel Methods (12 talks)
Applications (11 talks)
Preconditioners (7 talks)
Teoplitz and Circulant Matrix Solvers (3 talks)
Saddle Point Problem Solvers (3 talks)

In addition, two evening workshops were held on

Recent Progress in Iterative Software (5 talks)
Robust Iterative Methods (5 talks)

A special effort was made to bring students to the meeting. The vehicle for this was the Student Paper Competition, which was funded by the National Science Foundation. Students were asked to submit an original research paper consisting primarily of their own work and limited to 10 pages in length. The winners were chosen by a panel of judges taken from the program committee. Ten excellent papers were submitted, which made the judges' job difficult but gratifying. This year's winners were

First place: Johannas Tausch, Colorado State University, *"Equivalent Preconditioners for Boundary Element Methods"*;
Second place: Lina Hemmingsson, Uppsala University, *"Analysis of Semi-Toeplitz Preconditioners for First-Order PDE's"*;
Third place: Qing He, Arizona State University, *"Parallel Algorithms for Unconstrained Optimizations by Multisplitting."*

The winners presented their papers in a special session just prior to the conference banquet.

Due to the generosity of the NSF we were able to provide travel expenses, lodging, and registration for the three winners and lodging and registration for all contestants who were able to attend. Lodging and registration expenses were provided to a number of other students. In all, 22 students were given support.

This special issue contains a set of papers that represent a broad range of topics. The issue begins with a paper by Toh and Trefethen on fast calculation of pseudospectra of matrices, followed by a paper by Eisenstat and Walker on inexact Newton methods. Then, a group of six papers are concerned with preconditioned iterative methods. The first of this group, by Elman and Silvester, develops multilevel preconditionings for the Navier–Stokes equations that yield a condition independent of the mesh parameter. Also notable in this group are two student papers, one by Lina Hemmingsson and the other by Johannas Tausch. The next group of five papers involves multigrid methods either explicitly or as a preconditioning strategy. The first of this group, by Meza and Tuminaro, describes a multigrid preconditioner for semiconductor equations. This exemplifies the many talks at the meeting devoted to iterative methods applied to real world problems. The last paper in this group, by Henson, Limber, McCormick, and Robinson, develops a multilevel method for image reconstruction and is followed by two other papers on iterative solution of integral equations. The next two papers are concerned with parallel computing issues and are representative of the large number of talks on parallel computing at the meeting. The final three papers represent a cross section of new ideas in iterative methods.

I would like to thank the following members of the program committee for their help in editing this special issue.

Howard Elman
Anne Greenbaum
Seymour Parter
Homer Walker
Roland Freund
Steve McCormick
Nick Trefethen

Through their efforts, the articles contained in this special issue were carefully refereed and brought to print on schedule.

I would like to extend a special thanks to Fred Howes of the Applied Mathematical Sciences Program of the Department of Energy for generous support of this meeting. Without his help, this meeting could not have taken place. Special thanks also go to Mike Steuerwalt of the NSF Computational Mathematics Program for his generous support of the Student Paper Competition.

As this issue goes to press, planning for the next conference in this series is in its final stages. It will be held in Copper Mountain, Colorado, April 9–13, 1996 and will again focus on general iterative methods. It is our hope that the lively interaction and fine quality of presentations and papers that have marked previous meetings can be duplicated at the upcoming meeting.

Tom Manteuffel

SIAM J. SCI. COMPUT.
Vol. 17, No. 1, pp. 1–15, January 1996

001

CALCULATION OF PSEUDOSPECTRA BY THE ARNOLDI ITERATION*

KIM-CHUAN TOH† AND LLOYD N. TREFETHEN‡

Abstract. The Arnoldi iteration, usually viewed as a method for calculating eigenvalues, can also be used to estimate pseudospectra. This possibility may be of practical importance, because in applications involving highly nonnormal matrices or operators, such as hydrodynamic stability, pseudospectra may be physically more significant than spectra.

Key words. Arnoldi, Lanczos, pseudospectra, numerical range, hydrodynamic stability

AMS subject classifications. 65F15, 47A12

1. Introduction.

Large-scale nonsymmetric matrix eigenvalue problems, which typically arise via discretization of non-self-adjoint differential or integral operators, are commonly solved numerically by the Arnoldi iteration and its variants [1], [8], [18], [25]. In this paper we explore the possibility that the Arnoldi iteration can also be used for the estimation of pseudospectra. Such an idea was first proposed by Nachtigal, Reichel, and Trefethen [17] and Freund [7], and methods much closer to those of the present paper have been presented by Ruhe in talks at the Householder (1993) and St. Girons (1994) symposia on linear algebra [23]. Recent developments indicate that in some applications, the pseudospectra of a matrix or operator may be more significant physically than its spectrum (see §7). Since calculation of pseudospectra is much more expensive than calculation of spectra, this suggests that it may be desirable to develop methods for determining them iteratively.

The idea investigated here is that the pseudospectra of a matrix A can be approximated by those of the Hessenberg matrices constructed by an Arnoldi iteration. In the version of this paper originally submitted for publication [30], we made use of the $n \times n$ Hessenberg matrices H_n (see the next section for definitions). However, Tom Manteuffel recommended to us that it might be advantageous to consider instead the $(n+1) \times n$ Hessenberg matrices $\tilde{H}_n$, as is done, for example, in [13]. Meanwhile, this is also the idea that Axel Ruhe has been investigating. In the end we have decided to present experiments here for both H_n and $\tilde{H}_n$, while giving greater attention to the latter. For the examples we have computed, the distinction between the two makes little difference in practice, but $\tilde{H}_n$ has theoretical advantages (monotonic convergence) and also conceptual appeal, because it bypasses the usual consideration of Ritz values or "Arnoldi eigenvalue estimates." Thus we find ourselves exploring a new way of interpreting approximations not only of pseudospectra but also of spectra, one that may be of interest even in the special case of the Lanczos iteration for symmetric matrices.

For any $\epsilon \geq 0$, the ϵ-pseudospectrum of a matrix A is defined by

$$\Lambda_\epsilon(A) = \{z \in \mathbf{C} : \ \|(zI - A)^{-1}\| \geq \epsilon^{-1}\}, \tag{1}$$

with the convention $\|(zI - A)^{-1}\| = \infty$ if $z \in \Lambda(A)$, the spectrum of A [32]. If $\|\cdot\|$ is the 2-norm, an equivalent definition is

*Received by the editors May 18, 1994; accepted for publication (in revised form) April 10, 1995. This work was supported by NSF grant DMS-9116110 and DOE grant DE-FG02-94ER25199.

†Center for Applied Mathematics, Cornell University, Ithaca, NY 14853 (kc@cam.cornell.edu).

‡Department of Computer Science, Cornell University, Ithaca, NY 14853 (LNT@cs.cornell.edu).

$$\Lambda_\epsilon(A) = \{z \in \mathbf{C} : \sigma_{\min}(zI - A) \le \epsilon\}, \tag{2}$$

where $\sigma_{\min}(zI - A)$ denotes the smallest singular value of $zI - A$. Either definition makes it clear that the pseudospectra of a matrix are a family of nested subsets of C, with $\Lambda_0(A) = \Lambda(A)$. They can be computed most straightforwardly by evaluating $\sigma_{\min}(zI - A)$ on a grid of values in the complex z-plane, then sending the result to a contour plotter. S.-H. Lui has shown that this straightforward algorithm can be speeded up by a factor of five to ten by a preliminary reduction of A to Hessenberg form followed by inverse iteration combined with continuation [12]. Variants of this procedure involve triangular instead of Hessenberg reduction or Lanczos instead of inverse iteration. Such ideas apply to our matrices H_n and $\tilde{H}_n$ as well as A, and thus have little bearing on the relative speedup to be gained via Arnoldi iterations, so we shall not give details.

If A is normal (has a complete set of orthogonal eigenvectors), then $\Lambda_\epsilon(A)$ is just the closed ϵ-neighborhood of $\Lambda(A)$, but if A is far from normal, $\Lambda_\epsilon(A)$ may be much larger. These are the cases where difficulties are likely to arise if one tries to use the spectrum $\Lambda(A)$ to estimate quantities such as $\|A^n\|$, $\|e^{tA}\|$, or $\|f(A)\|$. Better estimates can often be obtained from the pseudospectra using methods such as Cauchy integrals, the Laplace transform, or the Kreiss matrix theorem [21], [32]–[34].

2. The Arnoldi iteration. Let A be a real or complex $m \times m$ matrix, and let $\|\cdot\|$ denote the 2-norm. A complete unitary reduction of A to upper Hessenberg form might be written $A = QHQ^*$ or $AQ = QH$. The idea of the Arnoldi iteration is to compute the successive steps of this reduction columnwise, starting from the condition that the first column of Q is a prescribed vector q_1 with $\|q_1\| = 1$. Let Q_n be the $m \times n$ matrix whose columns are the first n columns of Q,

$$Q_n = \left[\begin{array}{c|c|c|c} & & & \\ q_1 & q_2 & \cdots & q_n \\ & & & \end{array}\right], \tag{3}$$

and let $\tilde{H}_n$ be the $(n+1) \times n$ upper-left section of H,

$$\tilde{H}_n = \begin{pmatrix} h_{11} & & \cdots & h_{1n} \\ h_{21} & h_{22} & & \vdots \\ & \ddots & \ddots & \\ & & h_{n,n-1} & h_{nn} \\ & & & h_{n+1,n} \end{pmatrix}. \tag{4}$$

Then we have

$$AQ_n = Q_{n+1}\tilde{H}_n, \tag{5}$$

and the nth column of this equation can be written $Aq_n = h_{1n}q_1 + \cdots + h_{nn}q_n + h_{n+1,n}q_{n+1}$. The Arnoldi iteration is the modified Gram–Schmidt iteration that implements this $(n+1)$-term recurrence relation:

```
$q_1$ = arbitrary ($\|q_1\| = 1$)
for $n = 1, 2, 3, \ldots$
    $v = Aq_n$
    for $j = 1$ to $n$
        $h_{jn} = q_j^* v$
        $v = v - h_{jn} q_j$
    $h_{n+1,n} = \|v\|$
    $q_{n+1} = v / h_{n+1,n}$
```

The vectors $\{q_j\}$ form orthogonal bases of the successive *Krylov subspaces* generated by A and q_1,

$$\mathcal{K}_n = \langle q_1, Aq_1, \ldots, A^{n-1}q_1 \rangle = \langle q_1, q_2, \ldots, q_n \rangle \subseteq \mathbf{C}^m.$$

As a practical matter the iteration can be implemented with the aid of a "black box" procedure for computing the matrix–vector products Aq_n, which can be designed to take advantage of sparsity or other structure of A.

The product $Q_n^* Q_{n+1}$ is equal to the $n \times (n+1)$ identity, i.e., the $n \times (n+1)$ matrix with 1 on the main diagonal and 0 elsewhere. Therefore $Q_n^* Q_{n+1} \tilde{H}_n$ is the $n \times n$ Hessenberg matrix obtained by removing the last row of $\tilde{H}_n$,

$$H_n = \begin{pmatrix} h_{11} & & \cdots & h_{1n} \\ h_{21} & h_{22} & & \vdots \\ & \ddots & \ddots & \\ & & h_{n,n-1} & h_{nn} \end{pmatrix}. \tag{6}$$

From (5) we can accordingly derive the formula

$$H_n = Q_n^* A\, Q_n. \tag{7}$$

Though they differ in only one row, H_n and $\tilde{H}_n$ are entirely different objects. To highlight the difference, it is helpful to extend $\tilde{H}_n$ to the $m \times n$ matrix $\hat{H}_n$ consisting of the first n columns of H:

$$\hat{H}_n = \begin{pmatrix} h_{11} & & \cdots & h_{1n} \\ h_{21} & h_{22} & & \vdots \\ & \ddots & \ddots & \\ & & h_{n,n-1} & h_{nn} \\ & & & h_{n+1,n} \\ 0 & & & 0 \\ \vdots & & & \vdots \\ 0 & & & 0 \end{pmatrix}. \tag{8}$$

The matrices $\tilde{H}_n$ and $\hat{H}_n$ are identical except for the presence of $m - (n+1)$ additional rows of zeros in the latter, and thus they have, for example, the same rank, norm, and singular values.

We can interpret H_n and $\hat{H}_n$ as follows. With respect to the basis $\{q_1, q_2, \ldots\}$, $\hat{H}_n$ represents the action of A on $\mathcal{K}_n$, whereas H_n represents the same operation followed by an orthogonal projection back into $\mathcal{K}_n$. Since the domain and range of H_n are the same, it makes sense to speak of the eigenvalues of H_n; but since $\hat{H}_n$ has distinct domain and range, it does not make sense to speak of its eigenvalues.

3. Estimation of pseudospectra and numerical range. We propose that in many cases, for sufficiently large n, some of the pseudospectra of A can be reasonably approximated by the corresponding pseudospectra of H_n or $\tilde{H}_n$:

$$\Lambda_\epsilon(A) \approx \Lambda_\epsilon(H_n) \approx \Lambda_\epsilon(\tilde{H}_n). \tag{9}$$

For $n \ll m$, the computation of $\Lambda_\epsilon(H_n)$ or $\Lambda_\epsilon(\tilde{H}_n)$ will be $O((m/n)^3)$ times faster than that of $\Lambda_\epsilon(A)$. Note that in considering $\Lambda_\epsilon(\tilde{H}_n)$, we are dealing with the ϵ-pseudospectrum of a rectangular matrix. This set can be defined just as for square matrices by (1), with I now denoting a rectangular version of the identity and $(zI - A)^{-1}$ denoting the pseudoinverse. Equivalently, it can be defined by (2). As far as we know, pseudospectra of rectangular matrices have not been discussed before. However, the smallest singular value of $\tilde{H}_n$ has been considered in work by Meza, for example, on iterative solution of ill-conditioned systems of equations [16].

We are not aware of very satisfactory theorems to justify the approximation $\Lambda_\epsilon(A) \approx \Lambda_\epsilon(H_n)$. On the other hand the approximations $\Lambda_\epsilon(A) \approx \Lambda_\epsilon(\tilde{H}_n)$ converge monotonically (cf. (3.21) of [16] and Thm. 3.1 of [13]).

THEOREM 1. *Let an $m \times m$ matrix A be unitarily similar to a Hessenberg matrix H, and let $\tilde{H}_n$ denote the $(n+1) \times n$ section* (4). *(In particular, $\tilde{H}_n$ might be computed by an Arnoldi iteration, with arbitrary restarts in case a zero subdiagonal element $h_{n+1,n}$ is encountered.) Then for any z we have*

$$\sigma_{\min}(zI - \tilde{H}_1) \ge \sigma_{\min}(zI - \tilde{H}_2) \ge \sigma_{\min}(zI - \tilde{H}_3) \ge \cdots \ge \sigma_{\min}(zI - A), \tag{10}$$

and, consequently, for any $\epsilon \ge 0$,

$$\Lambda_\epsilon(\tilde{H}_1) \subseteq \Lambda_\epsilon(\tilde{H}_2) \subseteq \Lambda_\epsilon(\tilde{H}_3) \subseteq \cdots \subseteq \Lambda_\epsilon(A). \tag{11}$$

Proof. Since $zI - \tilde{H}_n$ and $zI - \hat{H}_n$ differ only by rows of zeros, they have the same singular values, and we may replace $\tilde{H}_n$ in (10) by $\hat{H}_n$. Since $zI - A$ and $zI - H$ are unitarily similar, they too have the same singular values, and we may replace A in (10) by H. Since $\hat{H}_n$ is simply the first n columns of H, (10) now follows directly from the characterization $\sigma_{\min}(A) = \min_{\|x\|=1} \|Ax\|$. By (2), this implies (11). □

It is interesting to compare Theorem 1 with the more familiar interpretation of Arnoldi and Lanczos iterations. Conventionally, a set of n Ritz values are considered at step n, and one is faced with the problem of estimating how close they may be to eigenvalues of A. In Theorem 1, there are no Ritz values. However, it may be noted that Ritz values can be defined as the points z at which $\sigma_{\min}(zI - H_n)$ achieves a local minimum (namely, zero). An analogue for the rectangular case would be to consider the points at which $\sigma_{\min}(zI - \tilde{H}_n)$ achieves a local minimum. It follows from (10) that this minimum value will be equal to zero if and only if z is an eigenvalue of A corresponding to an eigenvector that lies in the Krylov subspace $\mathcal{K}_n$.

Besides pseudospectra, it is well known that an Arnoldi iteration also may provide estimates of the numerical range (= field of values) of A, which we denote by $W(A)$. Now it is H_n that we most naturally make use of:

$$W(A) \approx W(H_n). \tag{12}$$

gain we have monotonic convergence (cf. Thm. 3.1 of [13]).

THEOREM 2. *Let A and H be as in Theorem* 1, *and let H_n denote the $n \times n$ section* (6). *Then*

$$W(H_1) \subseteq W(H_2) \subseteq W(H_3) \subseteq \cdots \subseteq W(A). \tag{13}$$

Proof. This is an easy consequence of the definition $W(A) = \{x^*Ax : \|x\| = 1\}$. □

4. Numerical experiments. We now turn to numerical examples computed in MATLAB. Consider first the $m \times m$ "Kahan matrix"

$$A = \begin{pmatrix} 1 & -c & -c & -c & -c \\ & s & -sc & -sc & -sc \\ & & s^2 & -s^2c & -s^2c \\ & & & \ddots & \vdots \\ & & & & s^{m-1} \end{pmatrix}, \tag{14}$$

where $s^{m-1} = 0.1$ and $s^2 + c^2 = 1$. Matrices of this type were proposed by Kahan to illustrate that QR factorization with column pivoting is not a fail-safe method of numerical rank determination [8], [11]. Pseudospectra of this matrix, for $m = 32$, were plotted in [32].

In Fig. 1, we take $m = 64$ and consider Arnoldi approximations $\Lambda_\epsilon(H_n)$ to $\Lambda_\epsilon(A)$ with $n = 5, 10, 15, 20$. At each of these steps, the figure shows the approximate numerical range and the ϵ-pseudospectra for $\epsilon = 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}$. In this and all of our numerical experiments, the initial vector q_1 is random (independent normally distributed entries).

To the eye, at least, the convergence of $\Lambda_\epsilon(H_n)$ to $\Lambda_\epsilon(A)$ in Fig. 1 is compelling. At $n = 5$, the Arnoldi iteration has not learned much of value, but by $n = 10$, recognizable approximations have begun to emerge. At $n = 20$ the approximations are excellent. Since the cost of an SVD grows cubically with the dimension of the matrix, and 1000 or more SVDs are involved in making one of these plots, calculations of pseudospectra even for these small matrices can be time-consuming, and if $n = 64$ can be replaced by $n = 20$, the savings will be a factor of around 30.

Note that, as is typical in cases of extreme nonnormality, the convergence of the eigenvalues of H_n to those of A in Fig. 1 is slow. The eigenvalues are too ill conditioned to be easily resolved. This is just the sort of problem where eigenvalues are likely to be of limited physical significance and where pseudospectra may provide a useful alternative. For a discussion of the physical significance of pseudospectra, including transient evolution phenomena, the effect of small perturbations, and the notion of "pseudo-resonance" in highly nonnormal systems, see [34].

Figure 2 shows the same computation, for the same matrix, except now the approximations are based on the rectangular Hessenberg matrix $\tilde{H}_n$. Broadly speaking, the approximations are about as good as in Fig. 1. Notice that there are no longer any Ritz values in the plot, which might be considered conceptually advantageous. On the other hand, in dealing with a matrix whose behavior was close to normal in some parts of the complex plane and far from normal in others, it might certainly be convenient to have Ritz values.

Not every matrix behaves as nicely as in Figs. 1 and 2. In Fig. 3 we consider the 64×64 "Grcar matrix," a Toeplitz matrix with -1 on the subdiagonal and 1 on the main diagonal and on the first three superdiagonals. This time, Arnoldi approximations based on $\tilde{H}_n$ at steps 10, 20, 30, and 40 are plotted, and only at $n = 40$ is reasonable convergence of the pseudospectra beginning to be evident. Although $(40/64)^3 \approx 0.24$ is still substantially less than 1, this is

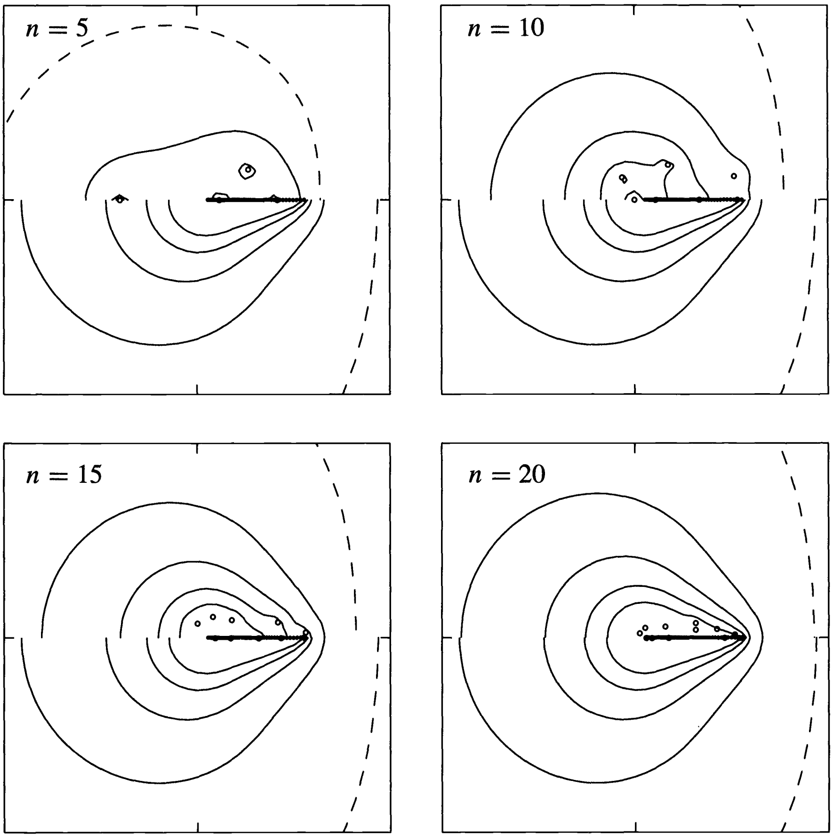


FIG. 1. *ϵ-pseudospectra of the* 64 × 64 *Kahan matrix* (14) *compared with those of four Arnoldi approximations* H_n ($\epsilon = 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}$). *The upper half of each plot corresponds to* $\Lambda_\epsilon(H_n)$, *and the lower half to* $\Lambda_\epsilon(A)$. *The dashed curves represent an analogous comparison of the numerical ranges* $W(H_n)$ *and* $W(A)$. *The small dots are the eigenvalues of* A *in the lower half-plane (hard to distinguish; they appear like a solid interval on the real axis), and the small circles are the eigenvalues of* H_n *in the upper half-plane (Ritz values). The real and imaginary axes are marked by ticks in each plot; the axis limits are* $-1.8 \leq \Re z \leq 1.8$, $-1.8 \leq \Im z \leq 1.8$.

a case where there is probably little to be gained in approximating $\Lambda_\epsilon(A)$ by $\Lambda_\epsilon(\tilde{H}_n)$ (or $\Lambda_\epsilon(H_n)$, whose performance is about the same).

Figures 2 and 3 represent two of the thirteen examples of highly nonnormal matrices considered in [32]. We have plotted Arnoldi approximations to pseudospectra for all of these and find that the Kahan matrix exhibits the best convergence and the Grcar matrix among the worst. Based on these examples alone, one would probably conclude that the Arnoldi approach to calculation of pseudospectra holds some promise but is not completely convincing.

However, the matrices of [32] are not typical of the large-scale problems that arise in practice. For these special matrices, selected for their dramatic pseudospectra, the nonnormality is such that all m eigenvalues are strongly coupled to one another. In applications, it is more typical for a small number of eigenvalues to be dominant and not strongly coupled to the others, with the behavior of the pseudospectra in the vicinity of these eigenvalues being of primary interest. In such cases Arnoldi approximations may perform better.

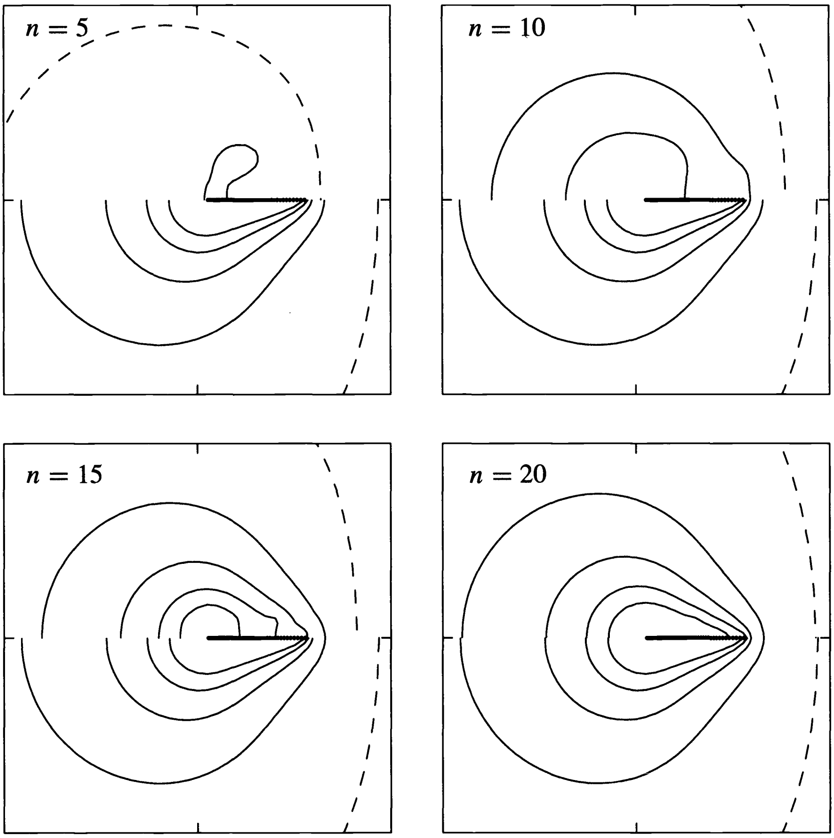


FIG. 2. *Same as Fig.* 1, *but now the curves in the upper half of each plot correspond to* $\Lambda_\epsilon(\tilde{H}_n)$ *instead of* $\Lambda_\epsilon(H_n)$. *Since the Ritz values have no simple connection to* $\tilde{H}_n$, *they are no longer shown. The dashed curves still correspond to* $W(H_n)$ *and* $W(A)$.

Figure 4 presents an example of this kind. Here A is the 64 × 64 bidiagonal matrix defined by

$$a_{k,k+1} = a_{k,k} = k^{-1/2}. \tag{15}$$

We can think of this as a prototype of a highly nonnormal compact operator and imagine that it is the behavior of the spectrum and pseudospectra away from the origin that is of interest. The convergence in this part of the complex plane is excellent. Moreover, it would be nearly as fast even if m were much larger than 64. For an example like this, the payoff in estimating pseudospectra iteratively could be huge.

5. Modified Arnoldi iterations. The large-scale matrices that arise most often in applications are of a kind different from all of our examples so far. These are discretizations of differential operators, which are not compact. As in Fig. 4, it is the behavior near a few leading eigenvalues that is of greatest interest, but the rest of the spectrum will typically extend to ∞ in the complex plane.

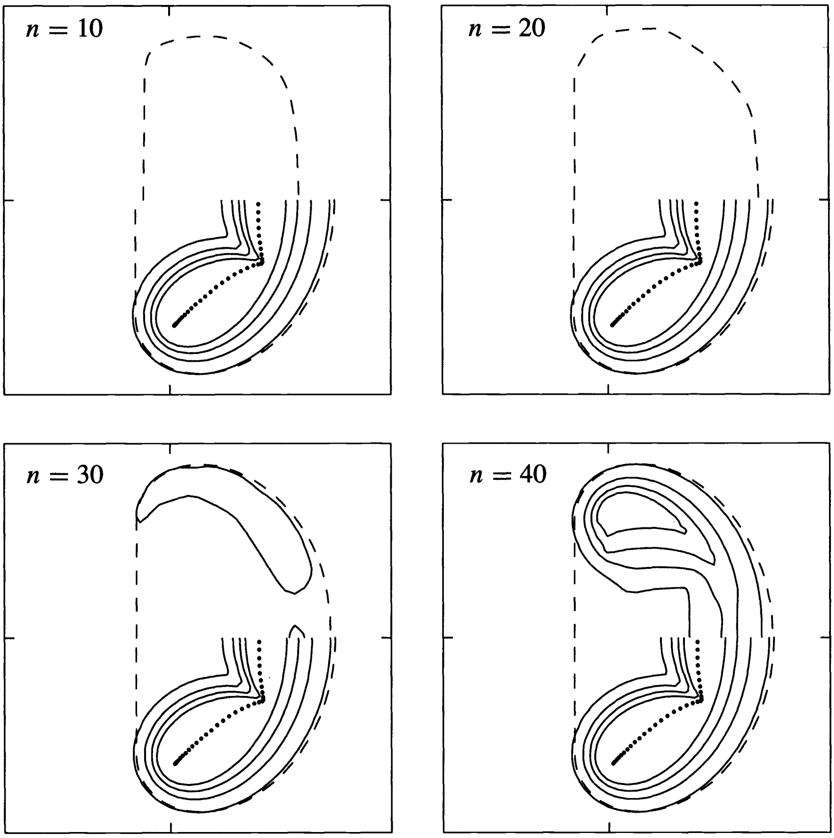


FIG. 3. *Same as Fig. 2, except for the pentadiagonal Grcar matrix of dimension $m = 64$. The axis limits are* $-3 \le \Re z \le 4$, $-3.5 \le \Im z \le 3.5$.

It is well known that a pure Arnoldi iteration may be ineffective in cases of this kind. As an example, consider another bidiagonal matrix defined by

$$a_{k,k} = -0.3k, \qquad a_{k,k+1} = 1. \tag{16}$$

Here the spectral and pseudospectral behavior near the origin should be largely unaffected by whether the dimension is 64 or 64,000. A pure Arnoldi iteration will have difficulty nonetheless, and the difficulty will increase with the dimension. This is illustrated in Fig. 5a, with $m = 64$, where we see quite disappointing convergence to the pseudospectra near $z = 0$.

Solutions to this problem have been proposed by a number of authors. One approach is to suppress the part of the spectrum far from the origin by an ancillary linear process such as a Chebyshev iteration or some other polynomial filter. Such ideas have been investigated by Chatelin and her colleagues and by Saad, Scott, Sorensen, and others [6], [9], [17], [24], [25], [27], [29]. A more powerful possibility, when it is feasible, is to modify the problem with the use of matrix inverses, in effect working with rational functions of A rather than just polynomials. Variations on this theme go by names such as shift-and-invert Arnoldi and rational Krylov iteration, and have been investigated by Ruhe, Saad, and Spence, among others [15], [22], [23], [25]. To be effective, such methods depend on the assumption that inverting A (i.e., solving a system $Ax = b$) is cheaper than the computation of main interest. This

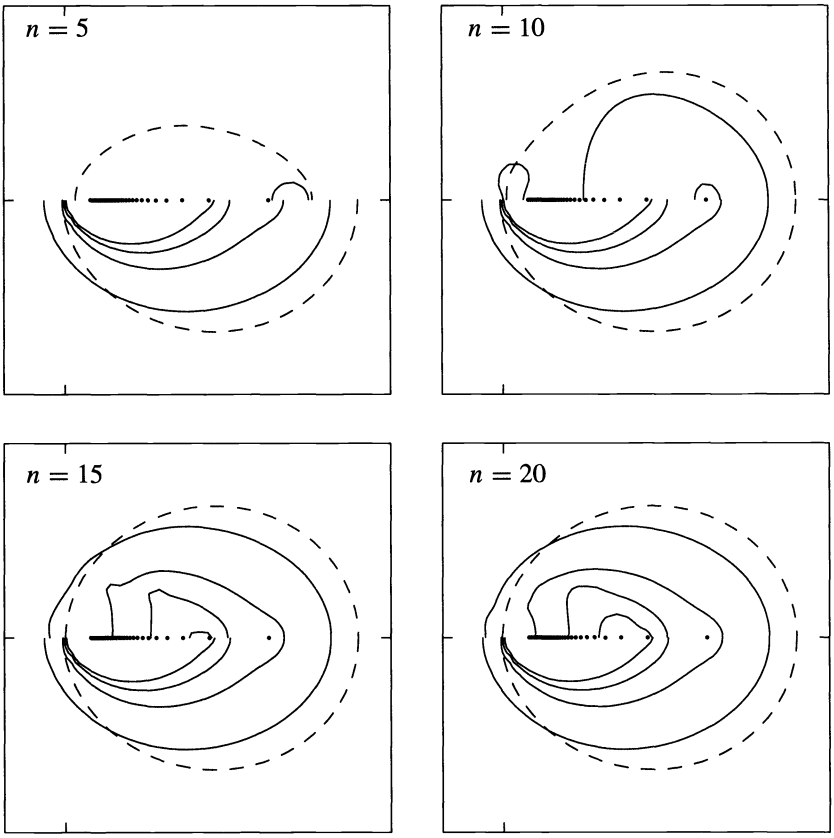


FIG. 4. *Same as Fig. 2 but for the bidiagonal matrix* (15), *a prototype of a compact operator. The axis limits are* $-0.3 \le \Re z \le 1.6$, $-0.95 \le \Im z \le 0.95$.

assumption is satisfied by many sparse eigenvalue problems, since sparsity can often be better taken advantage of for systems of equations than eigenvalues. It is amply satisfied in many calculations of pseudospectra, since these computations are even more expensive. Finally, an intermediate class of acceleration methods is based on preconditioning the eigenvalue problem by inverses not of A itself but of more easily inverted approximations $M \approx A$. This is the idea behind Davidson's method [4], [5] and related methods developed more recently by Meerbergen, Van der Vorst, and others [14], [25], [28].

We shall not attempt a systematic comparison of the uses for estimating pseudospectra of the various acceleration and preconditioning methods that have been proposed. Instead, we shall consider just the simplest modified Arnoldi process to give an idea of the great speedups that may be achieved by these methods. Figure 5b is a repetition of Fig. 5a in which the Arnoldi iteration has been replaced by an "inverse Arnoldi" iteration carried out with A^{-1} instead of A. This entails a solution of a system of equations involving A at each step, but this is a minor matter since A is bidiagonal. The result is a Hessenberg matrix H_n that approximates A^{-1}, and it is the pseudospectra of H_n^{-1} that we plot as approximations to those of A. Since H_n^{-1} is square, the plot also shows its eigenvalues.

Figure 5b shows excellent agreement of $\Lambda_\epsilon(H_n^{-1})$ and $\Lambda_\epsilon(A)$. Evidently the inverse-Arnoldi idea is highly effective for this problem. We take this as illustrative of the kind

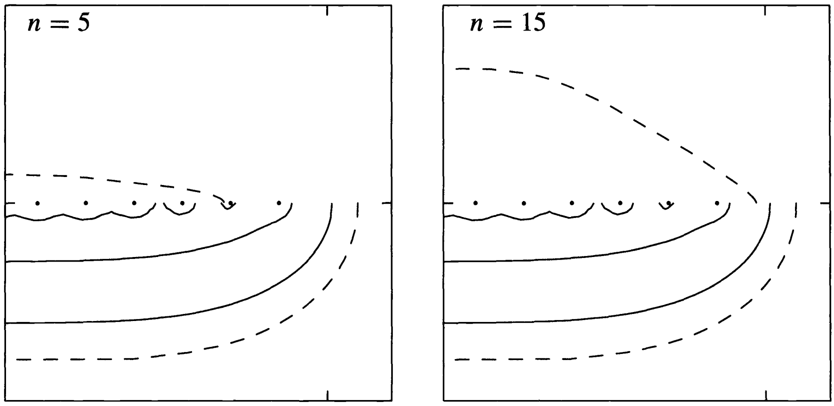


Fig. 5A. *Same as Fig.* 2 *but for the bidiagonal matrix* (16), *a prototype of an unbounded operator. The axis limits are* $-2 \le \Re z \le 0.4$, $-1.2 \le \Im z \le 1.2$.

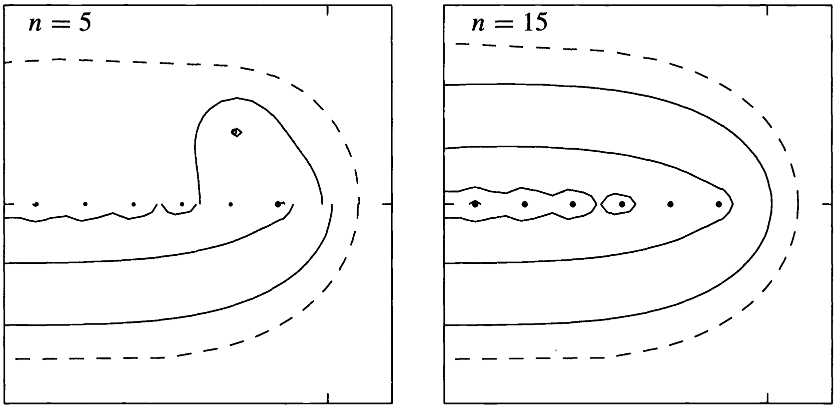


Fig. 5B. *Repetition of Fig.* 5a *with the Arnoldi iteration replaced by an inverse-Arnoldi iteration based on* A^{-1} *instead of* A. *The plot shows pseudospectra and eigenvalues of* H_n^{-1}.

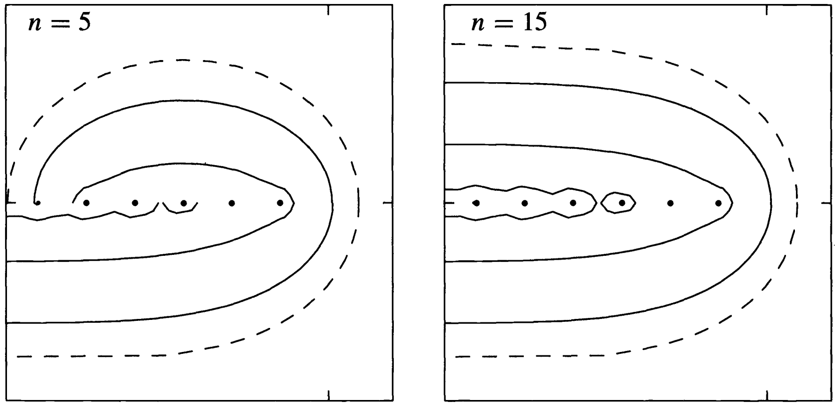


Fig. 5C. *Repetition of Fig.* 5a *with the Arnoldi iteration replaced by a projection of* A *onto the invariant subspace associated with the* n *eigenvalues of maximal real part.*

of gains that may be achieved by acceleration techniques in cases where A is invertible at reasonable cost. More sophisticated algorithms of this kind (based on $\tilde{H}_n$, not just H_n) are currently under development by Ruhe [23].

6. Projection onto an invariant subspace. There is another, more elementary, trick that we must not fail to mention. The calculation of pseudospectra is expensive, much more so than the calculation of a single eigenvalue decomposition. It follows that when the latter is affordable, much may be gained by simply calculating an eigenvalue decomposition of A, then projecting it onto the subspace of $\mathbf{C}^m$ spanned by certain eigenvectors. In typical applications these might be the eigenvectors associated with a subset of eigenvalues of A of maximal real part. This idea may be useful even when no Arnoldi iterations are in store; it is used, for example, in [20].

The mechanics of such a projection are as follows. We have already noted after (8) that if Q is an $m \times n$ matrix with orthonormal columns, then Q^*AQ represents the projection of A onto the column space of Q. Suppose now that we start with an $m \times n$ matrix V whose columns are selected eigenvectors of A, satisfying $AV = VD$ for some $n \times n$ diagonal eigenvalue matrix D. If $V = QR$ is a QR decomposition of V, with Q of dimension $m \times n$ and R of dimension $n \times n$, then we have $Q^*V = R$ and $Q = VR^{-1}$ and therefore

$$Q^*AQ = Q^*AVR^{-1} = Q^*VDR^{-1} = RDR^{-1}. \tag{17}$$

Thus RDR^{-1} (upper triangular) is the matrix representation of the projection of A onto the subspace spanned by the selected eigenvectors.

Figure 5c illustrates that for the example (16), this eigenvalue projection idea gives highly accurate pseudospectra. Despite its triviality, this trick can save a great deal of work. For example, an eigenvalue decomposition of a matrix of dimension 300 is a straightforward matter, whereas computing pseudospectra of such a matrix is a major project on most machines available today. If the dimension can be reduced to 30 by eigenvalue projection, the calculation of pseudospectra becomes easy.

Figure 6, following the format of Fig. 5, presents a less contrived example. Consider the convection-diffusion operator

$$\mathcal{L}u = u'' + u', \qquad u(0) = u(d) = 0 \tag{18}$$

acting on the interval $[0, d\,]$ (in the Hilbert space $L^2[0, d\,]$). The spectrum of this operator is a discrete, unbounded subset of the negative real axis, but, as discussed in [21], the pseudospectra are large regions in the left half-plane shaped approximately like parabolas. Taking $d = 40$, suppose we want to determine these pseudospectra in the neighborhood of the origin determined by the axis limits in Fig. 6. As discussed in [21], an efficient procedure is to construct a discretization matrix A based on Chebyshev spectral differentiation (we omit details). Unfortunately, for an accurate picture, the dimension of A has to be on the order of 100, making the calculation of the pseudospectra quite time-consuming, and if we wanted results in a larger region of the complex plane, matters would get worse. The figure shows that the inverse-Arnoldi idea works reasonably well here. Since A is dense, the eigenvalue projection idea is even better, and with $n = 40$ it produces a perfect picture with ten times less computing than would be involved in treating the full matrix with $m = 100$.

The dashed curves in Fig. 6, corresponding to the boundary of the numerical range, are worth a comment. Note that in Figs. 6b and 6c, we appear to have convincing convergence of the numerical range estimates in the upper half of the plot, but no dashed mirror image appears

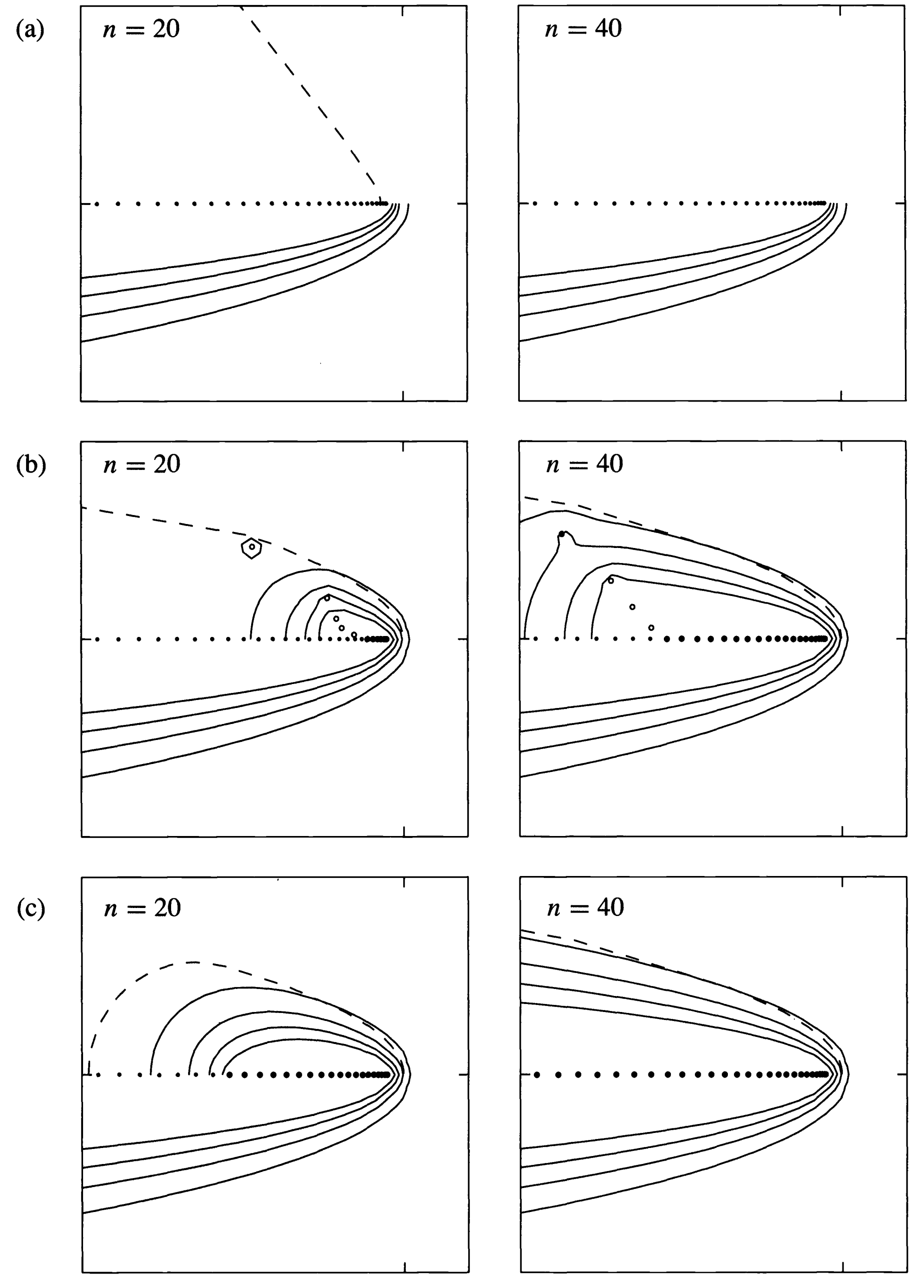


FIG. 6. *Same as Fig. 5, but for a* 100 × 100 *Chebyshev spectral approximation to the convection-diffusion operator* (18). *The axis limits are* $-5 \leq \Re z \leq 1$, $-3 \leq \Im z \leq 3$.

in the lower half. The explanation is that the actual 100 × 100 spectral differentiation matrix considered here has some huge "outlier" eigenvalues of size 1.2×10^4. These eigenvalues are artifacts of the discretization, with no relevance to the convection-diffusion operator $\mathcal{L}$, but they make the numerical range of A much larger than the axis scales of the figure. Thus the numerical range estimates in the upper half of Fig. 6 are no good at all, strictly speaking, for the matrix A being approximated. As it happens, however, they are excellent approximations to the numerical range of the operator $\mathcal{L}$. This stroke of good fortune is more than just coincidence, but of course one would have to be cautious about counting on such effects in applications.

For larger problems than those illustrated in this paper, a combination of the Arnoldi iteration and eigenvalue projections might be advantageous. A matrix of dimension 5000, for example, might be projected to dimension 100 by the Arnoldi iteration, whereupon an eigenvalue decomposition might be used to project further to a matrix of dimension 50, whose pseudospectra could then be plotted quickly by the methods proposed by Lui [12]. In such a sequence, the speedup over a naive calculation of pseudospectra might be on the order of many thousands.

7. Discussion. In this paper we have proposed that the iterative algorithms that have been developed for calculating spectra of large matrices may also be useful for estimating pseudospectra. If the matrix is a sparse approximation of a differential operator, the gains to be achieved by rational variants of the Arnoldi algorithm may be very great. If the matrix is dense, other acceleration devices may play a role, and surprisingly good results can be achieved by the simple method of computing the eigenvalue decomposition of A, then projecting onto an invariant subspace associated with a subset of the eigenvalues.

We have made no attempt to explain why our methods approximate pseudospectra as well as they do, beyond the lower bound on $\Lambda_\epsilon(A)$ of Theorem 1. An upper bound on $\Lambda_\epsilon(A)$ has been developed by Ruhe [23], but it appears to be far from sharp in practice.

It should be emphasized that the idea of using Arnoldi iterations for purposes more general than the calculation of eigenvalues is not new. The Arnoldi iteration potentially has relevance in all kinds of matrix problems where A is too big to deal with directly, but where there is reason to expect that the essential behavior can be captured by a low-dimensional projection. The example that has received the most attention is the use of Krylov subspaces to approximate e^{tA} [6], [10], [26]. Of course, the approximation of pseudospectra is not unrelated to the approximation of e^{tA}, since the ultimate purpose of estimating pseudospectra is often to obtain better insight into the behavior of $\|e^{tA}\|$ than the spectrum alone provides.

In closing, we would like to make two remarks.

Our first observation concerns the uses of Arnoldi iterations and the uses of spectra. The examples of this paper have shown that in cases of pronounced nonnormality, the behavior of a Krylov subspace iteration may be more closely tied to the pseudospectra of a matrix or operator than to its spectrum. A curious parallel of this statement is the recent discovery that in applications involving pronounced nonnormality, what is ultimately of physical interest may also be tied more closely to the pseudospectra than to the spectrum. In particular this is true of problems of hydrodynamic instability of fluid flows in a pipe or a channel, where traditional eigenvalue methods fail to explain the instabilities that are observed in practice, but pseudospectra do much better [2], [3], [19], [31], [34]. In our view, these two parallel statements about spectra and pseudospectra form a natural pair. In highly nonnormal problems, the Arnoldi iteration may indeed not be effective at determining eigenvalues, but we should not wish it to be. The information that it does acquire may be deeper and more valuable. It will be interesting to see whether this vision of broader uses of Arnoldi iterations in applications comes to fruition in upcoming years.

Our second remark is addressed to all those who use nonsymmetric Krylov subspace iterations or who produce software for such computations. For the present, due to the great variety of iterations and preconditioners that have been found to be useful, computations of this kind almost invariably involve an element of exploration. We believe that such explorations can be carried out far more effectively if the user habitually produces plots, not just numbers. Such plots might show Ritz values, lemniscates, pseudospectra of H_n, pseudospectra of $\tilde{H}_n$, eigenvalues of perturbed matrices, or other things—tastes differ, and at this time no one choice seems clearly superior to all others. But we firmly believe it is a mistake to plot nothing at all, or to plot nothing but Ritz values. The habit of looking at plots leads almost unconsciously to new questions and new understanding. Matrices and operators have personalities, which may be revealed with surprising economy by a few curves on the computer screen.

Acknowledgments. We are greatly indebted to Tom Manteuffel for his good advice, which has had a fundamental impact on this paper, and to Axel Ruhe for sharing with us his unpublished report [23]. We are also grateful to Anne Greenbaum for numerous technical suggestions.

REFERENCES

[1] W. E. Arnoldi, *The principle of minimized iterations in the solution of the matrix eigenvalue problem*, Quart. Appl. Math., 9 (1951), pp. 17–29.

[2] L. Boberg and U. Brosa, *Onset of turbulence in a pipe*, Z. Naturforschung, 43a (1988), pp. 697–726.

[3] K. M. Butler and B. F. Farrell, *Three-dimensional optimal perturbations in viscous shear flow*, Phys. Fluids A, 4 (1992), pp. 1637–1650.

[4] M. Crouzeix, B. Philippe, and M. Sadkane, *The Davidson method*, SIAM J. Sci. Comput., 15 (1994), pp. 62–76.

[5] E. R. Davidson, *The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices*, J. Comput. Phys., 17 (1975), pp. 87–94.

[6] W. S. Edwards, L. S. Tuckerman, R. A. Friesner, and D. C. Sorensen, *Krylov methods for the incompressible Navier-Stokes equations*, J. Comput. Phys., 110 (1994), pp. 82–102.

[7] R. W. Freund, *Quasi-kernel polynomials and their use in non-hermitian matrix iterations*, J. Comput. Appl. Math., 43 (1992), pp. 135–158.

[8] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 2nd ed., Johns Hopkins University Press, Baltimore, MD, 1989.

[9] D. Ho, *Tchebychev acceleration technique for large scale nonsymmetric matrices*, Numer. Math., 56 (1990), pp. 721–734.

[10] M. Hochbruck and C. Lubich, *On Krylov subspace approximations to the matrix exponential operator*, SIAM J. Numer. Anal., submitted.

[11] W. Kahan, *Numerical linear algebra*, Canad. Math. Bull., 9 (1966), pp. 757–801.

[12] S.-H. Lui, *Computation of pseudospectra by continuation*, SIAM J. Sci. Comput., 17 (1996), to appear.

[13] T. A. Manteuffel and G. Starke, *On hybrid iterative methods for nonsymmetric systems of linear equations*, Numer. Math., to appear.

[14] K. Meerbergen and D. Roose, *Matrix transformations for computing rightmost eigenvalues of large sparse nonsymmetric matrices*, Report TW206 (revised), Dept. of Comp. Sci., K.U. Leuven, April 1995.

[15] K. Meerbergen, A. Spence, and D. Roose, *Shift-invert and Cayley transforms for detection of rightmost eigenvalues of nonsymmetric matrices*, BIT, 34 (1994), pp. 409–423.

[16] J. C. Meza, *A Modification to the GMRES Method for Ill-Conditioned Linear Systems*, Report SAND95-8220, Sandia National Laboratories, Albuquerque, NM, April 1995.

[17] N. M. Nachtigal, L. Reichel, and L. N. Trefethen, *A hybrid GMRES algorithm for nonsymmetric linear systems*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 778–795.

[18] O. Nevanlinna, *Convergence of Iterations for Linear Equations*, Birkhäuser, Basel, 1993.

[19] S. C. Reddy and D. S. Henningson, *Energy growth in viscous channel flows*, J. Fluid Mech., 252 (1993), pp. 209–238.

[20] S. C. Reddy, P. J. Schmid, and D. S. Henningson, *Pseudospectra of the Orr–Sommerfeld operator*, SIAM J. Appl. Math., 53 (1993), pp. 15–47.

[21] S. C. Reddy and L. N. Trefethen, *Pseudospectra of the convection-diffusion operator*, SIAM J. Appl. Math., 54 (1994), pp. 1634–1649.

[22] A. Ruhe, *Rational Krylov algorithms for nonsymmetric eigenvalue problems*, II: *Matrix pairs*, Linear Algebra Appl., 197/198 (1994), pp. 283–296.

[23] ———, *The Rational Krylov Algorithm for Large Nonsymmetric Eigenvalues—Mapping the Resolvent Norms (Pseudospectrum)*, Draft report based on talk delivered at Sparse Days at St. Girons symposium (July, 1994), unpublished but available from `http://http.cs.berkeley.edu/~ruhe`, March, 1995.

[24] Y. Saad, *Chebyshev acceleration techniques for solving nonsymmetric eigenvalue problems*, Math. Comp., 42 (1984), pp. 567–588.

[25] ———, *Numerical Methods for Large Eigenvalue Problems*, Manchester U. Press, Manchester, UK, 1992.

[26] Y. Saad, *Analysis of some Krylov subspace approximations to the matrix exponential operator*, SIAM J. Numer. Anal., 29 (1992), pp. 209–228.

[27] J. A. Scott, *An Arnoldi code for computing selected eigenvalues of sparse real unsymmetric matrices*, Rep. RAL-93-097, Rutherford Appleton Lab., Oxon, England, December, 1993.

[28] G. L. G. Sleijpen and H. A. Van der Vorst, *A generalized Jacobi–Davidson iteration method for linear eigenvalue problems*, SIAM. J. Matrix Anal. Appl., 17 (1996), to appear.

[29] D. C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 375–385.

[30] K.-C. Toh and L. N. Trefethen, *Calculation of pseudospectra by the Arnoldi iteration*, Report CTC94TR179, Cornell Theory Center, Cornell University, Ithaca, NY, May, 1994.

[31] A. E. Trefethen, L. N. Trefethen, and P. J. Schmid, *Spectra and pseudospectra for pipe Poiseuille flow*, Report 94-16, Interdisciplinary Project Center for Supercomputing, Swiss Federal Inst. of Technology, Zurich, Switzerland, August, 1994.

[32] L. N. Trefethen, *Pseudospectra of matrices*, in Numerical Analysis 1991, D. F. Griffiths and G. A. Watson, eds., Longman Scientific and Technical, Harlow, UK, 1992.

[33] ———, *Spectra and Pseudospectra: The Behavior of Non-Normal Matrices and Operators*, manuscript.

[34] L. N. Trefethen, A. E. Trefethen, S. C. Reddy, and T. A. Driscoll, *Hydrodynamic stability without eigenvalues*, Science, 261 (1993), pp. 578–584.

SIAM J. SCI. COMPUT.
Vol. 17, No. 1, pp. 16–32, January 1996

CHOOSING THE FORCING TERMS IN AN INEXACT NEWTON METHOD*

STANLEY C. EISENSTAT[†] AND HOMER F. WALKER[‡]

Abstract. An inexact Newton method is a generalization of Newton's method for solving $F(x) = 0$, $F : \mathbb{R}^n \to \mathbb{R}^n$, in which, at the kth iteration, the step s_k from the current approximate solution x_k is required to satisfy a condition $\|F(x_k) + F'(x_k)s_k\| \le \eta_k\|F(x_k)\|$ for a "forcing term" $\eta_k \in [0, 1)$. In typical applications, the choice of the forcing terms is critical to the efficiency of the method and can affect robustness as well. Promising choices of the forcing terms are given, their local convergence properties are analyzed, and their practical performance is shown on a representative set of test problems.

Key words. forcing terms, inexact Newton methods, Newton iterative methods, truncated Newton methods, Newton's method, iterative linear algebra methods, GMRES

AMS subject classifications. 65H10, 65F10

1. Introduction.

Suppose that $F : \mathbb{R}^n \to \mathbb{R}^n$ is continuously differentiable in a neighborhood of $x_* \in \mathbb{R}^n$ for which $F(x_*) = 0$ and $F'(x_*)$ is nonsingular. Suppose further that F' is Lipschitz continuous at x_* with constant λ, i.e.,

$$\|F'(x) - F'(x_*)\| \le \lambda\|x - x_*\| \tag{1.1}$$

for x near x_*, where $\|\cdot\|$ denotes some norm on $\mathbb{R}^n$ and the induced norm on $\mathbb{R}^{n\times n}$.

An *inexact Newton method* (Dembo, Eisenstat, and Steihaug [4]) is an extension of classical Newton's method for approximating x_* formulated as follows:

Algorithm IN: Inexact Newton Method [4]

```
LET $x_0$ BE GIVEN.
FOR $k = 0$ STEP 1 UNTIL "CONVERGENCE" DO:
    FIND **some** $\eta_k \in [0, 1)$ AND $s_k$ THAT SATISFY
(1.2)        $\|F(x_k) + F'(x_k)s_k\| \le \eta_k\|F(x_k)\|$.
    SET $x_{k+1} = x_k + s_k$.
```

Note that (1.2) expresses both a certain reduction in the norm of $F(x_k) + F'(x_k)s$, the local linear model of F, and a certain accuracy in solving the *Newton equation* $F'(x_k)s = -F(x_k)$, the exact solution of which is the *Newton step.* In many applications, notably Newton iterative or truncated Newton methods[1], each η_k is specified first, and then an s_k is determined so that (1.2) holds. The role of η_k is, then, to force $\|F(x_k) + F'(x_k)s_k\|$ to be small in a particular way; accordingly, η_k is often called a *forcing term.*

The local convergence of an inexact Newton method is controlled by the forcing terms. Some specific illustrative results are the following (see Dembo, Eisenstat, and Steihaug [4]): Under the present assumptions, if x_0 is sufficiently close to x_* and $0 \le \eta_k \le \eta_{\max} < 1$ for each k, then $\{x_k\}$ converges to x_* q-linearly in the norm $\|\cdot\|_*$, defined by $\|v\|_* \equiv \|F'(x_*)v\|$ for

*Received by the editors June 1, 1994; accepted for publication (in revised form) March 15, 1995.

[†] Department of Computer Science, Yale University, P. O. Box 208285, New Haven, CT 06520-8285 (eisenstat-stan@cs.yale.edu). The research of this author was supported in part by U. S. Army Research Office contract DAAL03-91-G-0032.

[‡] Department of Mathematics and Statistics, Utah State University, Logan, UT 84322-3900 (walker@math.usu.edu). The work of this author was supported in part by United States Air Force Office of Scientific Research Grant AFOSR-91-0294, United States Department of Energy Grants DE-FG02-92ER25136 and DE-FG03-94ER25221, and National Science Foundation Grant DMS-9400217, all with Utah State University. It was done in part during a visit to the Computational and Applied Mathematics Department and the Center for Research on Parallel Computation, Rice University.

[1] These are implementations of Newton's method in which iterative linear algebra methods are used to solve the Newton equation approximately.

$v \in \mathbb{R}^n$, with asymptotic rate constant no greater than $\eta_{\max}$. Furthermore, if $\lim_{k\to\infty} \eta_k = 0$, then the convergence is q-superlinear, and if $\eta_k = O(\|F(x_k)\|)$, then the convergence is q-quadratic.[2]

In addition to controlling local convergence, there is another important issue associated with the forcing terms. Away from a solution, F and its local linear model may disagree considerably at a step that closely approximates the Newton step. Thus choosing η_k too small may lead to *oversolving* the Newton equation, by which we mean imposing an accuracy on an approximation of the Newton step that leads to significant disagreement between F and its local linear model. Oversolving may result in little or no decrease in $\|F\|$ and, therefore, little or no progress toward a solution. Moreover, in applications such as Newton iterative or truncated Newton methods, in which additional accuracy in solving the Newton equation requires additional expense, it may entail pointless costs; a less accurate approximation of the Newton step may be both cheaper and more effective.

Our purpose is to propose choices of the forcing terms that achieve desirably fast local convergence and also tend to avoid oversolving. All of the proposed choices incorporate information about F but are scale independent in that they do not change if F is multiplied by a constant.

In §2, we outline the proposed choices and analyze the local convergence of Algorithm IN that results from them; we also note some practical safeguards that improve performance. In §3, we discuss numerical experiments. The algorithm used in the experiments is a special case of Algorithm IN and is outlined in §3.1. The test problems are described in §3.2. An example of oversolving is given in §3.3, with additional observations and examples in §3.4. Summary test results are shown in §3.5. A summary discussion is given in §4.

Preliminaries. We define some useful constants and formulate several elementary results. Set $M \equiv \max\left\{\|F'(x_*)\|, \|F'(x_*)^{-1}\|\right\}$. For $\delta > 0$, define

$$N_\delta(x_*) \equiv \{x \in \mathbb{R}^n : \|x - x_*\| < \delta\},$$

and let $\delta_* > 0$ be sufficiently small that

1. F is continuously differentiable and F' is nonsingular on $N_{\delta_*}(x_*)$,
2. $\|F'(x)^{-1}\| \le 2M$ for $x \in N_{\delta_*}(x_*)$,
3. inequality (1.1) holds for $x \in N_{\delta_*}(x_*)$,
4. $\delta_* < 2/(\lambda M)$.

LEMMA 1.1. *If $x \in N_{\delta_*}(x_*)$ and if s is such that $x_+ \equiv x + s \in N_{\delta_*}(x_*)$, then*

$$\|F(x_+) - F(x) - F'(x)s\| \le \lambda\left(2\|x - x_*\| + \frac{\|s\|}{2}\right)\|s\|.$$

Proof. Setting $x(t) \equiv x + ts$ for $0 \le t \le 1$, we have

$$\begin{aligned}
\|F(x_+) - F(x) - F'(x)s\| &= \left\|\int_0^1 F'(x(t))\,s\,dt - F'(x)\,s\right\| \\
&\le \left\|\int_0^1 \left[F'(x(t)) - F'(x_*)\right]dt - \left[F'(x) - F'(x_*)\right]\right\| \|s\| \\
&\le \left(\int_0^1 \lambda\left[\|x - x_*\| + t\|s\|\right]dt + \lambda\|x - x_*\|\right)\|s\| \\
&= \lambda\left(2\|x - x_*\| + \frac{\|s\|}{2}\right)\|s\|. \qquad \square
\end{aligned}$$

[2] See, e.g., Dennis and Schnabel [6, §§2.3 and 3.1] for definitions of the types of convergence referred to throughout this paper.

LEMMA 1.2. *There is a* $\mu > 0$ *such that*

$$\frac{1}{\mu}\|x - x_*\| \le \|F(x)\| \le \mu\|x - x_*\|$$

whenever $x \in N_{\delta_*}(x_*)$.

Proof. With Lemma 1.1, we have

$$\begin{aligned}\|F(x)\| &\le \|F'(x_*)(x - x_*)\| + \|F(x) - F(x_*) - F'(x_*)(x - x_*)\| \\ &\le M\|x - x_*\| + \frac{\lambda}{2}\|x - x_*\|^2 \le \left(M + \frac{\lambda\delta_*}{2}\right)\|x - x_*\|\end{aligned}$$

and

$$\begin{aligned}\|F(x)\| &\ge \|F'(x_*)(x - x_*)\| - \|F(x) - F(x_*) - F'(x_*)(x - x_*)\| \\ &\ge \frac{1}{M}\|x - x_*\| - \frac{\lambda}{2}\|x - x_*\|^2 \ge \left(\frac{1}{M} - \frac{\lambda\delta_*}{2}\right)\|x - x_*\|.\end{aligned}$$

The lemma follows with $\mu \equiv \max\left\{M + \lambda\delta_*/2, (1/M - \lambda\delta_*/2)^{-1}\right\}$. □

LEMMA 1.3. *If* $x \in N_{\delta_*}(x_*)$ *and* $\|F(x) + F'(x)\,s\| \le \eta\|F(x)\|$ *for some* s *and* $\eta \in [0, 1)$, *then* $\|s\| \le 4M\|F(x)\|$.

Proof. We have

$$\begin{aligned}\|s\| &\le \|F'(x)^{-1}\|\,\|F'(x)\,s\| \\ &\le 2M\big(\|F(x)\| + \|F(x) + F'(x)\,s\|\big) \\ &\le 2M(1 + \eta)\|F(x)\| \le 4M\|F(x)\|. \quad \square\end{aligned}$$

LEMMA 1.4. *There is a* $B > 0$ *such that if* $x \in N_{\delta_*}(x_*)$, s *and* $\eta \in [0, 1)$ *are such that* $\|F(x) + F'(x)\,s\| \le \eta\|F(x)\|$, *and* $x_+ \equiv x + s \in N_{\delta_*}(x_*)$, *then*

$$\|F(x_+)\| \le \big(\eta + B\|F(x)\|\big)\|F(x)\|.$$

Proof. With Lemmas 1.1–1.3, we have that

$$\begin{aligned}\|F(x_+)\| &\le \|F(x) + F'(x)\,s\| + \|F(x_+) - F(x) - F'(x)\,s\| \\ &\le \eta\|F(x)\| + \lambda\left(2\|x - x_*\| + \frac{\|s\|}{2}\right)\|s\| \\ &\le \eta\|F(x)\| + \lambda\big(2\mu\|F(x)\| + 2M\|F(x)\|\big)\cdot 4M\|F(x)\| \\ &= \big(\eta + B\|F(x)\|\big)\|F(x)\|,\end{aligned}$$

where $B \equiv 8\lambda M(\mu + M)$. □

2. The proposed choices. In the analysis in this section, we use the Lipschitz constant λ in (1.1) and the constants M, δ_*, μ, and B introduced in the preliminaries in §1. We also let δ be such that $0 < \delta \le \delta_*/(1 + 4\mu M)$ and note the following consequence of Lemmas 1.2 and 1.3.

PROPOSITION 2.1. *If $x \in N_\delta(x_*)$ and $\|F(x) + F'(x)s\| \le \eta\|F(x)\|$ for some s and $\eta \in [0, 1)$, then $x + s \in N_{\delta_*}(x_*)$.*

We assume for convenience that Algorithm IN continues indefinitely without termination and that $F(x_k) \neq 0$ for all k. Note that if $x_k \in N_{\delta_*}(x_*)$, then $F'(x_k)$ is nonsingular and, therefore, suitable s_k and x_{k+1} exist for any $\eta_k \in [0, 1)$. Our standing assumptions on F and x_* are those made in the first paragraph of §1.

Our first choice is the following.

Choice 1: Given $\eta_0 \in [0, 1)$, choose

$$\eta_k = \frac{\|F(x_k) - F(x_{k-1}) - F'(x_{k-1})s_{k-1}\|}{\|F(x_{k-1})\|}, \qquad k = 1, 2, \ldots, \tag{2.1}$$

or

$$\eta_k = \frac{\Big|\|F(x_k)\| - \|F(x_{k-1}) + F'(x_{k-1})s_{k-1}\|\Big|}{\|F(x_{k-1})\|}, \qquad k = 1, 2, \ldots. \tag{2.2}$$

Note that η_k given by either (2.1) or (2.2) directly reflects the agreement between F and its local linear model at the previous step. The choice (2.2) may be more convenient to evaluate than (2.1) in some circumstances. Since it is at least as small, local convergence will be at least as fast as with (2.1); however, if it is significantly smaller, then it may be more difficult to find a suitable step in some applications and perhaps risk greater oversolving as well.

THEOREM 2.2. *Under the standing assumptions on F and x_*, if x_0 is sufficiently near x_*, then $\{x_k\}$ produced by Algorithm IN with $\{\eta_k\}$ given by Choice 1 remains in $N_{\delta_*}(x_*)$ and converges to x_* with*

$$\|x_{k+1} - x_*\| \le \beta\|x_k - x_*\|\|x_{k-1} - x_*\|, \qquad k = 1, 2, \ldots, \tag{2.3}$$

for a constant β independent of k.

Remark. It follows immediately from (2.3) that the convergence is q-superlinear and two-step q-quadratic. As in the case of the classical secant method, it also follows that the convergence is of r-order $(1 + \sqrt{5})/2$; see, e.g., Stoer and Bulirsch [14, p. 293] for the argument.

Proof. It suffices to prove the theorem with $\{\eta_k\}$ given by (2.1).

Suppose that $\eta_0 \in [0, 1)$ is given. Let τ be such that $\eta_0 < \tau < 1$, and let $\epsilon > 0$ be sufficiently small that $\eta_0 + B\epsilon \le \tau$, $\big[8\lambda M(\mu + M) + B\big]\epsilon \le \tau$, and $\epsilon < \delta/\mu$. Note that if $x \in N_{\delta_*}(x_*)$ and $\|F(x)\| \le \epsilon$, then $x \in N_\delta(x_*)$ by Lemma 1.2.

Let $x_0 \in N_\delta(x_*)$ be sufficiently near x_* that $\|F(x_0)\| \le \epsilon$. Since $x_0 \in N_\delta(x_*)$, we have $x_1 \in N_{\delta_*}(x_*)$ by Proposition 2.1. Also, by Lemma 1.4,

$$\begin{aligned} \|F(x_1)\| &\le \big(\eta_0 + B\|F(x_0)\|\big)\|F(x_0)\| \le \big(\eta_0 + B\epsilon\big)\|F(x_0)\| \\ &\le \tau\|F(x_0)\| \le \|F(x_0)\| \le \epsilon, \end{aligned} \tag{2.4}$$

and, hence, $x_1 \in N_\delta(x_*)$.

As an inductive hypothesis, suppose that, for some $k \ge 1$, we have $x_k \in N_\delta(x_*)$, $x_{k-1} \in N_\delta(x_*)$, $\|F(x_k)\| \le \epsilon$, and $\|F(x_{k-1})\| \le \epsilon$. Then $x_{k+1} \in N_{\delta_*}(x_*)$ by Proposition 2.1, and

Lemmas 1.1–1.3 give

$$\begin{aligned}\eta_k &= \frac{\|F(x_k) - F(x_{k-1}) - F'(x_{k-1})\, s_{k-1}\|}{\|F(x_{k-1})\|} \\ &\le \frac{\lambda\big(2\|x_{k-1} - x_*\| + \|s_{k-1}\|/2\big)\|s_{k-1}\|}{\|F(x_{k-1})\|} \\ &\le \frac{\lambda\big(2\mu\|F(x_{k-1})\| + 2M\|F(x_{k-1})\|\big)\cdot 4M\|F(x_{k-1})\|}{\|F(x_{k-1})\|} \\ &= 8\lambda M(\mu + M)\|F(x_{k-1})\|.\end{aligned}$$

Then Lemma 1.4 implies

$$\begin{aligned}\|F(x_{k+1})\| &\le \big(\eta_k + B\|F(x_k)\|\big)\|F(x_k)\| \\ &\le \Big[8\lambda M(\mu + M)\|F(x_{k-1})\| + B\|F(x_k)\|\Big]\|F(x_k)\| \\ &\le \big[8\lambda M(\mu + M) + B\big]\epsilon\|F(x_k)\| \le \tau\|F(x_k)\|.\end{aligned} \tag{2.5}$$

Thus $\|F(x_{k+1})\| \le \tau\|F(x_k)\| \le \epsilon$ and, hence, $x_{k+1} \in N_\delta(x_*)$.

It follows from this induction that $\{x_k\} \subset N_\delta(x_*) \subset N_{\delta_*}(x_*)$. Furthermore, (2.4) and (2.5) give $\|F(x_{k+1})\| \le \tau\|F(x_k)\|$ for each $k \ge 0$; hence, $F(x_k) \to 0$ and, by Lemma 1.2, $x_k \to x_*$ as well.

To show (2.3), we note that (2.4) and (2.5) give, for $k \ge 1$, $\|F(x_k)\| \le \|F(x_{k-1})\|$ and

$$\begin{aligned}\|F(x_{k+1})\| &\le \Big[8\lambda M(\mu + M)\|F(x_{k-1})\| + B\|F(x_k)\|\Big]\|F(x_k)\| \\ &\le \big[8\lambda M(\mu + M) + B\big]\|F(x_{k-1})\|\,\|F(x_k)\|.\end{aligned}$$

With Lemma 1.2, this implies (2.3) with $\beta \equiv \mu^3\big[8\lambda M(\mu + M) + B\big]$. □

One possible way to obtain faster local convergence while retaining the potential advantages of (2.1) and (2.2) is to raise those expressions to powers greater than one. A particular possibility that we considered in our numerical experiments is squaring those expressions. We note without proof that this leads to local convergence with

$$\|x_{k+1} - x_*\| \le \beta \max\left\{\|x_{k-1} - x_*\|^2, \|x_k - x_*\|\right\}\|x_k - x_*\|, \quad k = 1, 2, \ldots,$$

which implies that $x_k \to x_*$ r-quadratically. However, this possibility was not as successful in our experiments as the other choices proposed here, and we do not consider it further.

Our second choice is the following.

Choice 2: Given $\gamma \in [0, 1]$, $\alpha \in (1, 2]$, and $\eta_0 \in [0, 1)$, choose

$$\eta_k = \gamma\left(\frac{\|F(x_k)\|}{\|F(x_{k-1})\|}\right)^\alpha, \quad k = 1, 2, \ldots. \tag{2.6}$$

The choice (2.6) does not directly reflect the agreement between F and its local linear model, as does Choice 1. However, the experiments in §3 show that it results in little oversolving in practice, and the following theorem shows that it offers attractive local convergence.

THEOREM 2.3. *Under the standing assumptions on F and x_*, if x_0 is sufficiently near x_*, then $\{x_k\}$ produced by Algorithm IN with $\{\eta_k\}$ given by Choice 2 remains in $N_{\delta_*}(x_*)$ and converges to x_*. If $\gamma < 1$, then the convergence is of q-order α. If $\gamma = 1$, then the convergence is of r-order α and of q-order p for every $p \in [1, \alpha)$.*

Proof. Suppose that $\eta_0 \in [0, 1)$ is given and let $\epsilon > 0$ be sufficiently small that $\eta_0 + B\epsilon \leq \eta_0^{1/\alpha}$ and $\epsilon < \delta/\mu$. Note that if $x \in N_{\delta_*}(x_*)$ and $\|F(x)\| \leq \epsilon$, then $x \in N_\delta(x_*)$ by Lemma 1.2.

Let $x_0 \in N_\delta(x_*)$ be sufficiently near x_* that $\|F(x_0)\| \leq \epsilon$. As an inductive hypothesis, suppose that, for some $k \geq 0$, we have $x_k \in N_\delta(x_*)$, $\|F(x_k)\| \leq \epsilon$, and $\eta_k \leq \eta_0$. Since $x_k \in N_\delta(x_*)$, we have $x_{k+1} \in N_{\delta_*}(x_*)$ by Proposition 2.1. Also, by Lemma 1.4,

$$\begin{aligned}\|F(x_{k+1})\| &\leq \big(\eta_k + B\|F(x_k)\|\big)\|F(x_k)\| \\ &\leq \big(\eta_0 + B\epsilon\big)\|F(x_k)\| \leq \eta_0^{1/\alpha}\,\|F(x_k)\|.\end{aligned} \tag{2.7}$$

Then $\|F(x_{k+1})\| \leq \eta_0^{1/\alpha}\,\epsilon \leq \epsilon$, and it follows that $x_{k+1} \in N_\delta(x_*)$. Furthermore, (2.7) gives

$$\eta_{k+1} = \gamma\big(\|F(x_{k+1})\|/\|F(x_k)\|\big)^\alpha \leq \gamma\eta_0 \leq \eta_0.$$

It follows from this induction that $\{x_k\} \subset N_\delta(x_*) \subset N_{\delta_*}(x_*)$. Furthermore, (2.7) gives $\|F(x_{k+1})\| \leq \eta_0^{1/\alpha}\,\|F(x_k)\|$ for each $k \geq 0$; hence, $F(x_k) \to 0$ and, by Lemma 1.2, $x_k \to x_*$ as well.

It remains to show the desired rates of convergence. Note that, for $k > 0$, (2.7) and (2.6) give

$$\|F(x_{k+1})\| \leq \left[\gamma\left(\frac{\|F(x_k)\|}{\|F(x_{k-1})\|}\right)^\alpha + B\|F(x_k)\|\right]\|F(x_k)\|. \tag{2.8}$$

First, suppose that $\gamma < 1$ and set $\rho_k \equiv \|F(x_k)\|/\|F(x_{k-1})\|^\alpha$ for $k > 0$. From (2.8) and (2.7), we have $\rho_{k+1} \leq \gamma\rho_k + B\|F(x_k)\|^{2-\alpha} \leq \gamma\rho_k + B\|F(x_0)\|^{2-\alpha}$ for $k > 0$, and it follows inductively that

$$\rho_{k+1} \leq \gamma^k\rho_1 + \left(\sum_{j=0}^{k-1}\gamma^j\right)B\|F(x_0)\|^{2-\alpha} \leq \rho_1 + \frac{B}{1-\gamma}\|F(x_0)\|^{2-\alpha}.$$

Thus $\{\rho_k\}$ is uniformly bounded. Consequently, $F(x_k) \to 0$ with q-order α, and it follows from Lemma 1.2 that $x_k \to x_*$ with q-order α as well.

Now, suppose that $\gamma = 1$. We first show that the convergence is of q-order p for $p \in [1, \alpha)$. For $k > 0$, (2.8) gives

$$\begin{aligned}\|F(x_{k+1})\| &\leq \left[\left(\frac{\|F(x_k)\|}{\|F(x_{k-1})\|}\right)^\alpha + B\|F(x_k)\|\right]\|F(x_k)\| \\ &= \left[\left(\frac{\|F(x_k)\|}{\|F(x_{k-1})\|}\right)^{\alpha-p}\frac{\|F(x_k)\|}{\|F(x_{k-1})\|^p} + B\|F(x_k)\|^{2-p}\right]\|F(x_k)\|^p.\end{aligned} \tag{2.9}$$

For each $k > 0$, set $\sigma_k \equiv \|F(x_k)\|/\|F(x_{k-1})\|^p$ and recall that (2.7) gives $\|F(x_k)\| \leq \eta_0^{1/\alpha}\,\|F(x_{k-1})\|$, whence $\|F(x_k)\| \leq \eta_0^{k/\alpha}\,\|F(x_0)\|$. Then for $k > 0$, (2.9) implies

$$\sigma_{k+1} \leq \eta_0^{1-p/\alpha}\sigma_k + B\eta_0^{k(2-p)/\alpha}\|F(x_0)\|^{2-p} \leq \xi\sigma_k + \xi^k C,$$

where $\xi \equiv \eta_0^{1-p/\alpha}$ and $C \equiv B\|F(x_0)\|^{2-p}$. It follows inductively that

$$\sigma_{k+1} \le \xi^k \left(\sigma_1 + kC\right),$$

and, hence,

$$\|F(x_{k+1})\| \le \xi^k \left(\sigma_1 + kC\right) \|F(x_k)\|^p.$$

Since $\xi^k \left(\sigma_1 + kC\right) \to 0$ as $k \to \infty$, we conclude that $F(x_k) \to 0$ with q-order p and, by Lemma 1.2, $x_k \to x_*$ with q-order p as well.

Still assuming $\gamma = 1$, we now show that $x_k \to x_*$ with r-order α. By Lemma 1.2, it suffices to show that $\|F(x_k)\| \to 0$ with r-order α; we shall prove the somewhat stronger result that $\tau_k \equiv \|F(x_k)\|/\|F(x_{k-1})\| \to 0$ with r-order α.

It follows from the results above that $\tau_k \to 0$. Then there is a k_0 such that $(2\tau_{k_0+1})^{(\alpha-1)} + 2B\|F(x_{k_0})\| \le 1$. For convenience, we re-index if necessary so that $k_0 = 0$. Then $(2\tau_1)^{(\alpha-1)} + 2B\|F(x_0)\| \le 1$, which implies $D \equiv 1/(2\tau_1) > 1$. Set $\beta_k \equiv D\tau_k$ for $k \ge 0$. Note that $\beta_1 = 1/2$. It suffices to show that $\beta_k \to 0$ with r-order α.

We claim that $\beta_k \le \beta_1^{\alpha^{k-1}}$ for $k = 1, 2, \ldots$, from which it follows that $\beta_k \to 0$ with r-order α. The claim clearly holds for $k = 1$. Suppose that it holds up to some $k \ge 1$. Then Lemma 1.4 implies

$$\|F(x_{k+1})\| \le \left(\tau_k^\alpha + B\|F(x_k)\|\right) \|F(x_k)\|,$$

whence

$$\tau_{k+1} \le \tau_k^\alpha + B\tau_k \ldots \tau_1 \|F(x_0)\|.$$

From this, we obtain

$$\begin{aligned}
\beta_{k+1} &\le D^{1-\alpha}\beta_k^\alpha + \frac{B\|F(x_0)\|}{D^{k-1}}\beta_k \cdots \beta_1 \\
&\le D^{1-\alpha}\left(\beta_1^{\alpha^{k-1}}\right)^\alpha + B\|F(x_0)\|\beta_1^{(\alpha^{k-1}+\cdots+1)} \\
&\le \left(D^{1-\alpha} + B\|F(x_0)\|/\beta_1\right)\beta_1^{\alpha^k} \\
&= \left[(2\tau_1)^{\alpha-1} + 2B\|F(x_0)\|\right]\beta_1^{\alpha^k} \le \beta_1^{\alpha^k},
\end{aligned}$$

and the proof is complete. □

It is possible to show local convergence for Algorithm IN when $\{\eta_k\}$ is given by Choice 2 with $\gamma > 1$, provided η_0 is sufficiently small. However, Choice 2 with $\gamma > 1$ was not competitive in our experiments.

2.1. Practical safeguards. Although the forcing term choices given above are usually effective in avoiding oversolving, we have observed in experiments that they occasionally become too small far away from a solution. There is a particular danger of the Choice 1 forcing terms becoming too small; indeed, an η_k given by (2.1) or (2.2) can be undesirably small because of either a very small step or coincidental very good agreement between F and its local linear model. In our experiments, we observed relatively few occasions on which the Choice 2 forcing terms became undesirably small; however, this did occur.

We introduce safeguards here that are intended to prevent the forcing terms from becoming too small too quickly. The rationale is that if large forcing terms are appropriate at some point, then subsequent forcing terms should not be allowed to become much smaller until this has been justified over several iterations. These are not claimed to be the most effective safeguards that might be devised for general use or even for the test problems used in our experiments. However, they were consistently effective in our tests, more so than several other possibilities that we tried, and they serve to demonstrate the usefulness of safeguards.

For each choice, we restrict η_k to be no less than a certain minimum value, but only if that minimum value is above a threshhold. The minimum value is determined by raising η_{k-1} to a power associated with the rate of convergence of the (unsafeguarded) choice. The threshhold that we use here is .1; this is clearly somewhat arbitrary but was effective in our experiments. Note that, in each case, the minimum value eventually drops below the threshhold whenever there is convergence to a solution. Thus the safeguards eventually become inactive whenever there is convergence, and the asymptotic convergence is that for the unsafeguarded choice given by the theorems above.

For Choice 1, the safeguard is the following:

Choice 1 *safeguard:* Modify η_k by $\eta_k \leftarrow \max\{\eta_k, \eta_{k-1}^{(1+\sqrt{5})/2}\}$ whenever $\eta_{k-1}^{(1+\sqrt{5})/2} > .1$.

For perspective, recall from the remark after Theorem 2.2 that the convergence of (2.3) implies convergence of r-order $(1+\sqrt{5})/2$. For Choice 2, the safeguard is the following:

Choice 2 *safeguard:* Modify η_k by $\eta_k \leftarrow \max\{\eta_k, \gamma\eta_{k-1}^{\alpha}\}$ whenever $\gamma\eta_{k-1}^{\alpha} > .1$.

Finally, we note that, away from a solution, it may be possible for each of the proposed choices to be greater than one. Accordingly, it may be necessary in practice to impose an additional safeguard to make sure that $\eta_k \in [0, 1)$ for each k, as in the algorithm in §3.1 below that was used in our experiments.

3. Numerical experiments. In this section, we report on numerical experiments with the forcing term choices outlined in §2, modified with the safeguards given in §2.1. In the experiments, for computational convenience, we always used η_k given by (2.2) for Choice 1. For Choice 2, we used $\gamma = 1, .9, .5$ and $\alpha = 2, (1+\sqrt{5})/2$. The latter value of α results in an order of convergence roughly comparable to that for Choice 1; see Theorem 2.3 and the remark after Theorem 2.2. For a broader comparison, we also included the following representative forcing term choices:

1. the choice $\eta_k = 10^{-1}$, which requires modestly accurate approximations of Newton steps and results in local q-linear convergence in the norm $\|\cdot\|_*$.
2. the choice $\eta_k = 10^{-4}$ used by Cai, Gropp, Keyes, and Tidriri [3], which requires uniformly close approximations of Newton steps for all k and results in fast local q-linear convergence in the norm $\|\cdot\|_*$.
3. the choice $\eta_k = 1/2^{k+1}$ of Brown and Saad [2]. This choice results in local q-superlinear convergence and allows relatively inaccurate approximations of Newton steps for small k, when x_k may not be near x_*; however, it incorporates no information about F.
4. the choice $\eta_k = \min\{1/(k+2), \|F(x_k)\|\}$ of Dembo and Steihaug [5]. This choice results in q-quadratic local convergence and also may allow relatively inaccurate approximations of Newton steps for small k. It incorporates some information about F; however, it does not reflect the agreement of F and its local linear model and, in addition, depends on the scale of F.

3.1. The algorithm. A globalized inexact Newton algorithm was necessary because initial approximate solutions were not always near a solution. We used Algorithm INB of

Eisenstat and Walker [7, §6]. This is an inexact Newton method globalized by backtracking, which we write here as follows.

Algorithm INB: Inexact Newton Backtracking Method [7]

```
LET $x_0$, $\eta_{\max} \in [0, 1)$, $t \in (0, 1)$, AND $0 < \theta_{\min} < \theta_{\max} < 1$ BE GIVEN.
FOR $k = 0$ STEP 1 UNTIL "CONVERGENCE" DO:
    CHOOSE AN **initial** $\eta_k \in [0, \eta_{\max}]$ AND $s_k$ SUCH THAT
        $\|F(x_k) + F'(x_k)\, s_k\| \leq \eta_k \|F(x_k)\|$.
    WHILE $\|F(x_k + s_k)\| > [1 - t(1 - \eta_k)]\|F(x_k)\|$ DO:
        CHOOSE $\theta \in [\theta_{\min}, \theta_{\max}]$.
        UPDATE $s_k \longleftarrow \theta s_k$ AND $\eta_k \longleftarrow 1 - \theta(1 - \eta_k)$.
    SET $x_{k+1} = x_k + s_k$.
```

Note that Algorithm INB requires $\eta_k \in [0, \eta_{\max}]$ for each initial η_k. For the safeguarded choices in §2, this necessitates the additional safeguard $\eta_k \leftarrow \min\{\eta_k, \eta_{\max}\}$.

Theorem 6.1 of Eisenstat and Walker [7] states that if $\{x_k\}$ generated by Algorithm INB has a limit point x_* such that $F'(x_*)$ is invertible, then $F(x_*) = 0$ and $x_k \rightarrow x_*$. Furthermore, in this case, the initial η_k and s_k are accepted without modification for all sufficiently large k; it follows in particular that the asymptotic convergence to x_* is determined by the initial η_k's.

In implementing Algorithm INB, we first chose each initial η_k (with $\eta_0 = 1/2$ for Choices 1 and 2) and then determined an initial s_k by approximately solving the Newton equation using GMRES(m), the restarted GMRES method of Saad and Schultz [12], with restart value $m = 20$. Products of $F'(x_k)$ with vectors were evaluated analytically in some cases and approximated by finite differences of F-values in others; see §3.2. When finite-difference approximations were used, a second-order central difference was used to evaluate the initial residual at the beginning of each cycle of 20 GMRES steps, and subsequently first-order forward differences were used within the cycle. This selective second-order differencing gave essentially the same accuracy as if central differences had been used throughout, but at much lower cost (see Turner and Walker [16]).

The parameters used were $\eta_{\max} = .9$, $t = 10^{-4}$, $\theta_{\min} = 1/10$, and $\theta_{\max} = 1/2$. The norm was the Euclidean norm $\|\cdot\|_2$. In the while-loop, each θ was chosen to minimize over $[\theta_{\min}, \theta_{\max}]$ the quadratic $p(\theta)$ for which $p(0) = g(0)$, $p'(0) = g'(0)$, and $p(1) = g(1)$, where $g(\theta) \equiv \|F(x_k + \theta s_k)\|_2^2$. Convergence was declared when either $\|F(x_k)\|_2 \leq 10^{-12}\|F(x_0)\|_2$ or $\|s_k\|_2 \leq 10^{-12}$. These tight stopping tolerances allowed asymptotic convergence behavior to become evident.[3] Failure was declared when one of the following occurred: (1) k reached 200 without convergence, (2) an initial s_k was not found in 1000 GMRES(20) iterations, or (3) ten iterations of the while-loop failed to produce an acceptable step. All computing was done in double precision on Sun Microsystems workstations using the Sun Fortran compiler.

3.2. The test problems.

The test set consisted of four PDE problems and two integral equation problems. The PDE problems are all elliptic boundary value problems posed on $\Omega \equiv [0, 1] \times [0, 1] \subseteq \mathbb{R}^2$.

3.2.1. A PDE problem.

The problem is

$$\Delta u + u^3 = 0 \text{ in } \Omega, \qquad u = 0 \text{ on } \partial\Omega.$$

[3] In some applications, less stringent convergence tolerances are commonly used. As a result, asymptotic convergence behavior may not be very important, and it may be appropriate to use forcing terms that are not asymptotically increasingly demanding, such as constant forcing terms that give adequately fast q-linear convergence.

This problem has multiple solutions, but only one that is positive everywhere (McKenna [10], Schaaf [13]). These properties appear to be shared by the discretized problem, and finding the everywhere-positive solution can be difficult without a good initial approximate solution. Discretization was by the usual centered differences on a 100 × 100 uniform grid, so that $n = 10^4$. The discretized problem was preconditioned on the right using a fast Poisson solver from FISHPACK (Swartztrauber and Sweet [15]). Products of F' with vectors were evaluated analytically. The initial approximate solution was a discretization of $u_0(x) \equiv \kappa x_1(1 - x_1)x_2(1 - x_2)$, which should lead to the everywhere-positive solution for large κ. Two test cases were considered: $\kappa = 100$ and $\kappa = 1000$. For the latter value, the initial approximate solution is farther from the solution and the problem is harder.

3.2.2. The (modified) Bratu problem. The problem is

$$\Delta u + \kappa \frac{\partial u}{\partial x_1} + \lambda e^u = 0 \text{ in } \Omega, \qquad u = 0 \text{ on } \partial\Omega.$$

The actual Bratu (or Gelfand) problem has $\kappa = 0$; see, e.g., Glowinski, Keller, and Reinhart [8] or the description by Glowinski and Keller in the collection of nonlinear model problems assembled by Moré [11, pp. 733–737]. As κ and λ grow, solving the Newton equations for the discretized problem becomes harder for GMRES(20). Discretization and preconditioning were as in §3.2.1. Products of F' with vectors were evaluated analytically. The initial approximate solution was zero. Two test cases were considered: $\kappa = \lambda = 10$ and $\kappa = \lambda = 20$.

3.2.3. The driven cavity problem. The problem is

$$(1/Re)\Delta^2\psi + \frac{\partial\psi}{\partial x_1}\frac{\partial}{\partial x_2}\Delta\psi - \frac{\partial\psi}{\partial x_2}\frac{\partial}{\partial x_1}\Delta\psi = 0 \quad \text{in } \Omega,$$

$$\psi = 0 \quad \text{and} \quad \frac{\partial\psi}{\partial n} = g \quad \text{on } \partial\Omega,$$

where $g(x_1, x_2) = 1$ if $x_2 = 1$ and $g(x_1, x_2) = 0$ if $0 \leq x_2 < 1$. This is a widely used test problem; see, e.g., Brown and Saad [2] or Glowinski, Keller, and Reinhart [8]. The numerical problem becomes harder as the Reynolds number Re increases. Discretization was by piecewise-linear finite elements on a uniform 63 × 63 grid[4], so that $n = 3969$. The discretized problem was preconditioned on the right using a fast biharmonic solver of Bjørstad [1]. Products of F' with vectors were approximated with finite differences. The initial approximate solution was zero. Two test cases were considered: $Re = 100$ and $Re = 500$.

3.2.4. The porous medium equation. The problem considered here is

$$\Delta\left(u^2\right) + d\frac{\partial}{\partial x_1}(u^3) + f = 0 \quad \text{in } \Omega,$$

with $u = 1$ on the bottom and left sides of Ω and $u = 0$ on the top and right sides. This is more or less a steady-state special case of a general problem considered by van Duijn and de Graaf [17]. Discretization was by the usual centered differences on a 64 × 64 uniform grid, so that $n = 4096$. The discretized problem was preconditioned on the right using the tridiagonal part of the Jacobian. Products of F' with vectors were evaluated analytically. The function f was a point source of magnitude 50 at the lower left grid point. The initial approximate solution was a discretization of $u_0(x) \equiv 1 - x_1x_2$ on the interior grid points, which tended to require more backtracking for negative d and to cause more oversolving for positive d. Two test cases were considered: $d = 50$ and $d = -50$.

[4] We thank P. N. Brown for providing the code for this.

3.2.5. An integral equation. The problem, from Kelley and Northrup [9], is

$$cu(x)^2 - \frac{1}{2}\int_0^1 \cos(yu(x))u(y)\,dy + \frac{1}{2}\sin 1 - c = 0, \quad x \in [0,1].$$

Clearly, $u(x) \equiv 1$ is always a solution, and there exist other solutions for at least some values of c. The discretized problem was determined by approximating integrals using 20-point Gaussian quadrature[5] over 20 subintervals of $[0, 1]$, so that $n = 400$. No preconditioning was necessary. Products of F' with vectors were approximated with finite differences. The initial approximate solution was a discretization of $u_0(x) \equiv 1 + \kappa \cos 9\pi x$. One test case was considered: $c = \kappa = 1.25$.

3.2.6. The Chandrasekhar H-equation. The problem is

$$u(x) - \frac{1}{1 - Lu(x)} = 0, \quad x \in [0,1],$$

where

$$Lu(x) \equiv \frac{c}{2}\int_0^1 \frac{xu(\xi)}{x+\xi}\,d\xi.$$

This problem arises in radiative transfer problems; see, e.g., the description by Kelley in the Moré problem collection [11, pp. 737–739]. The continuous problem is singular at $c = 1$, and so is the discretized problem considered here with discretization as in §3.2.5. The discretized problem becomes more difficult to solve as $c \to 1$ but is still tractable at $c = 1$. As in §3.2.5, no preconditioning was necessary. Products of F' with vectors were approximated with finite differences. The initial approximate solution was zero. Three test cases were considered: $c = .5$, $c = .999$, and $c = 1$.

3.3. An example of oversolving. Algorithm INB with the Dembo–Steihaug [5] choice $\eta_k = \min\{1/(k+2), \|F(x_k)\|_2\}$ was applied to the driven cavity problem with $Re = 500$. The results are shown in Fig. 3.1, in which the logarithms of the norms of F and its local linear model are plotted as dotted and solid curves, respectively, versus the numbers of GMRES(20) iterations. (Most of the F-values used for Figs. 3.1–3.4 would not normally be available but were computed for these illustrations.) Triangles indicate the start of new inexact Newton steps. In this example, $\eta_k = \|F(x_k)\|_2$ for each $k > 0$; the safeguard value $\eta_k = 1/(k+2)$ was never invoked for $k > 0$.

In Fig. 3.1, gaps between the solid and dotted curves indicate oversolving. Note that once oversolving begins, there is virtually no further reduction in $\|F\|_2$ until the beginning of the next inexact Newton step; thus further GMRES(20) iterations represent wasted effort. Note also the vertical discontinuity in the dotted curve at the end of the fourth inexact Newton step (after 45 GMRES(20) iterations); this indicates a reduction of the initial inexact Newton step through backtracking.

To show the benefits gained by reducing oversolving, we applied Algorithm INB with η_k given by the safeguarded Choice 1 to the same problem. The results are shown in Fig. 3.2. Note that oversolving is almost eliminated and there are no step reductions through backtracking. Also, the total number of GMRES(20) iterations is 221, compared to 327 in the previous case. However, the number of inexact Newton steps is 12, compared to 10 previously.

[5] We thank C. T. Kelley for providing the code for this.

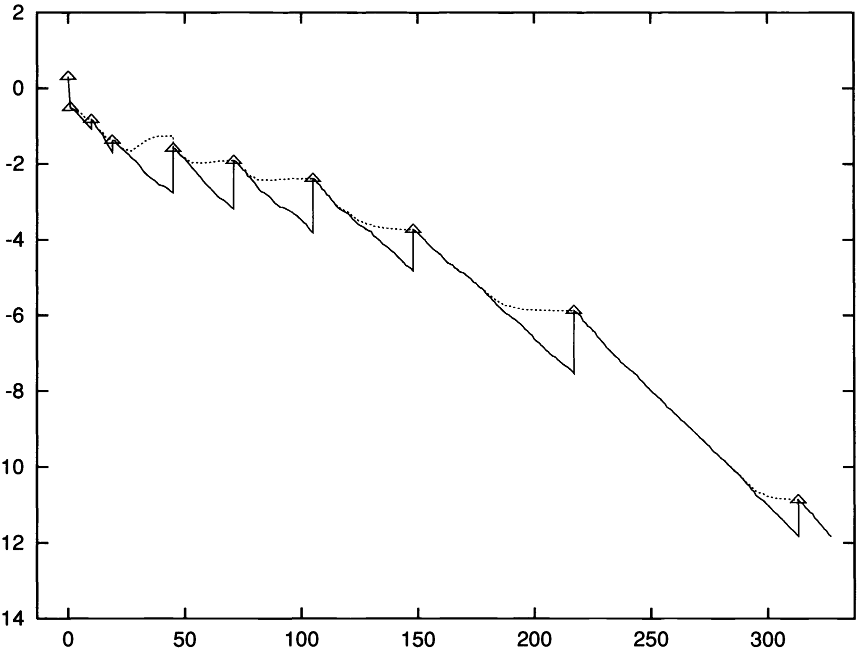


FIG. 3.1. *Illustration of oversolving with* $\eta_k = \min\{1/(k+2), \|F(x_k)\|_2\}$ *on the driven cavity problem with* $Re = 500$. *The horizontal axis indicates the number of* GMRES(20) *iterations. The solid curve is* $\log_{10} \|F + F's\|_2$; *the dotted curve is* $\log_{10} \|F\|_2$. *Triangles indicate new inexact Newton steps.*

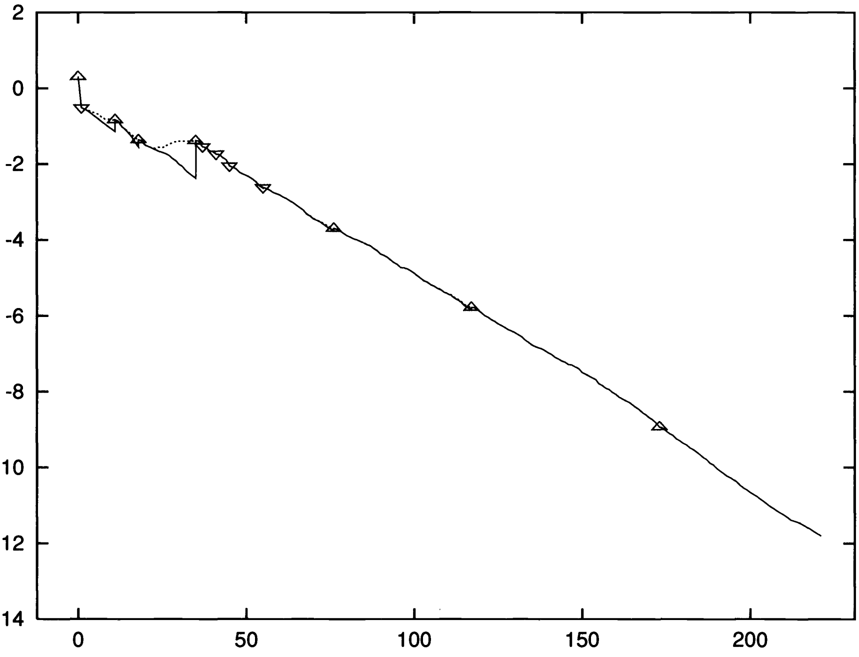


FIG. 3.2. *Illustration of reduction of oversolving with the safeguarded Choice* 1 *forcing terms on the driven cavity problem with* $Re = 500$. *The horizontal axis indicates the number of* GMRES(20) *iterations. The solid curve is* $\log_{10} \|F + F's\|_2$; *the dotted curve is* $\log_{10} \|F\|_2$. *Triangles indicate new inexact Newton steps: "*Δ*" indicates* η_k *given by* (2.2); *"*∇*" indicates the safeguard value.*

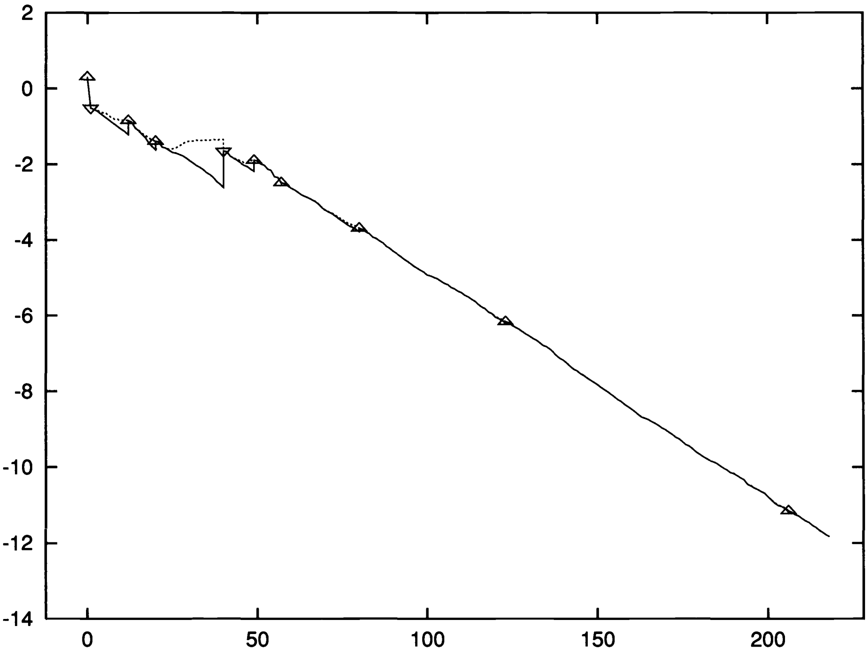


FIG. 3.3. *Illustration of the performance of Algorithm INB with selective second-order differencing and safeguarded Choice* 2 *forcing terms,* $\alpha = 2$, $\gamma = .9$, *on the driven cavity problem with* $Re = 500$. *The horizontal axis indicates the number of* GMRES(20) *iterations. The solid curve is* $\log_{10} \|F + F's\|_2$; *the dotted curve is* $\log_{10} \|F\|_2$. *Triangles indicate new inexact Newton steps: "*Δ*" indicates* η_k *given by* (2.6); "∇" *indicates the safeguard value.*

3.4. Additional observations and examples. In an algorithm such as the implementation of Algorithm INB used here, choosing a very small forcing term may risk more than needless expense in obtaining an unnecessarily accurate solution of the Newton equation. First, if oversolving results, then disagreement between F and its local linear model may require significant work from the globalization procedure or even cause it to fail. In the example in §3.3, the choice $\eta_k = \min\{1/(k+2), \|F(x_k)\|_2\}$ required one backtracking, while the safeguarded Choice 1 did not. We observed a more dramatic example involving the PDE problem of §3.2.1 with $\kappa = 1000$. With the safeguarded Choice 1, the iterates from Algorithm INB converged to the everywhere-positive solution in 40 GMRES(20) iterations; two backtracks were required. With the choice $\eta_k = \min\{1/(k+2), \|F(x_k)\|_2\}$, 164 GMRES(20) iterations and 11 backtracks were necessary; furthermore, convergence was to a solution other than the everywhere-positive solution. Such convergence to a "wrong" solution may or may not be undesirable per se, but it does indicate the potentially serious effects of disagreement between F and its local linear model.

Second, unless special care is taken, a very small forcing term may require more residual reduction than an iterative linear solver such as GMRES can accurately deliver, especially when products of F' with vectors are approximated with finite differences. Recall from §3.1 that our implementation of Algorithm INB uses selective second-order differencing to obtain essentially the same accuracy as if second-order differences were used throughout. Using the safeguarded Choice 2 forcing terms with $\alpha = 2$ and $\gamma = .9$, we applied this implementation to the driven cavity problem with $Re = 500$; the results are shown in Fig. 3.3. There is no evidence of inaccuracy in GMRES(20), and 218 iterations were required for successful termination. However, when the implementation was changed to use only first-order forward

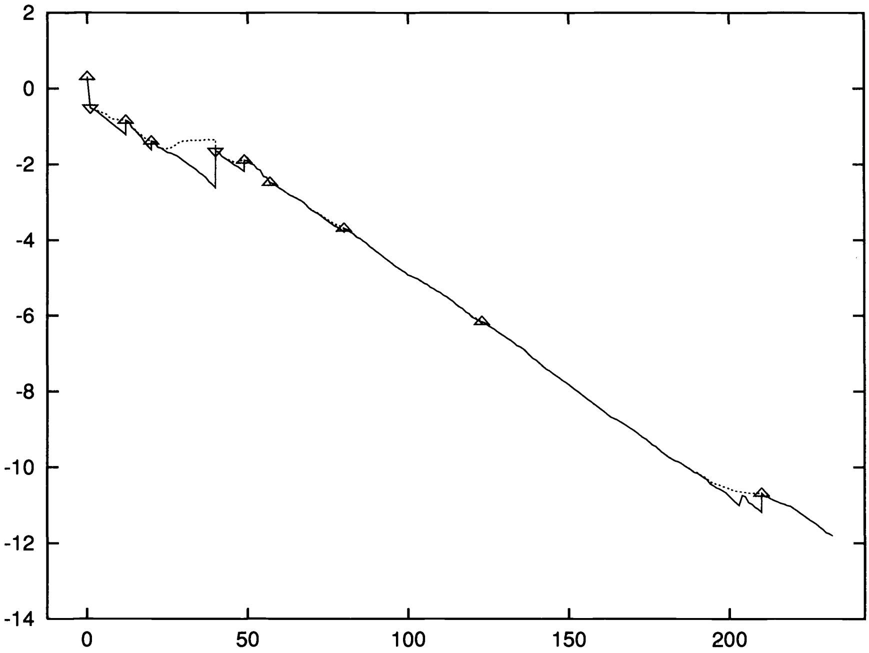


FIG. 3.4. *Illustration of the performance of Algorithm INB with first-order differencing throughout and safeguarded Choice* 2 *forcing terms,* $\alpha = 2$, $\gamma = .9$, *on the driven cavity problem with* $Re = 500$. *The horizontal axis indicates the number of* GMRES(20) *iterations. The solid curve is* $\log_{10} \|F + F's\|_2$; *the dotted curve is* $\log_{10} \|F\|_2$. *Triangles indicate new inexact Newton steps: "*Δ*" indicates* η_k*, given by* (2.6); *"*∇*" indicates the safeguard value.*

differences throughout, we obtained the results in Fig. 3.4. Note the increase in the linear residual norm curve (the solid curve) just after iteration 200. The linear residual norm values used for this curve were evaluated directly at the beginning of each GMRES(20) cycle and then maintained recursively within the cycle; the observed increase occurs after the direct evaluation at iteration 200 and indicates that the recursively maintained values have become inaccurate. We note also that the number of GMRES(20) iterations required for termination has increased to 232.

3.5. Summary test results. In Table 3.1, we summarize the results of applying Algorithm INB to all test problem cases described in §3.2. In Table 3.2, we summarize the results over the PDE problem cases only. The results for the PDE problems are broken out in a separate table not only because these problems constitute an important problem class but also because the characteristic performance of Algorithm INB on these problems differed from that on the integral equations. On the integral equations, and on the H-equation in particular, GMRES(20) was so effective that the effects of different forcing term choices tended to be obscured. In most cases, only one to three GMRES(20) iterations were required for each inexact Newton step, and the linear residual norm was often reduced by several orders of magnitude in a single iteration. On the PDE problems, many more GMRES(20) iterations were typically required for each inexact Newton step, with only modest linear residual norm reduction per GMRES(20) iteration. Thus the PDE problems gave a somewhat more refined view of the effects of different forcing term choices.

The first three columns of Tables 3.1 and 3.2 give geometric means of the numbers of linear iterations (GMRES(20) iterations), inexact Newton steps, and "function evaluation equivalents," where, for each test case, we define the number of "function evaluation

TABLE 3.1

Summary test results over all problems. GMLI, GMINS, *and* GMFEE *are geometric means of the numbers of linear iterations, inexact Newton steps, and "function evaluation equivalents," respectively.* NB, NW, *and* NFAIL *are the total numbers of backtracks, instances of convergence to a "wrong" solution, and failures, respectively. Results marked "*" were over successful runs only.*

η_k choice	GMLI	GMINS	GMFEE	NB	NW	NFAIL
10^{-1}	65.5*	12.00*	82.3*	2*	1*	1
10^{-4}	90.2*	7.21*	103.3*	1*	0*	2
$1/2^{k+1}$	70.3*	9.24*	85.4*	6*	1*	1
$\min\{1/(k+2), \Vert F(x_k)\Vert_2\}$	72.2	8.72	86.5	18	2	0
Choice 1	51.7	9.14	65.3	5	0	0
Choice 2, $\alpha = 2, \gamma = 1$	51.8	8.38	64.3	6	0	0
Choice 2, $\alpha = 2, \gamma = .9$	52.5	7.89	64.7	8	0	0
Choice 2, $\alpha = 2, \gamma = .5$	66.8	7.93	79.4	13	1	0
Choice 2, $\alpha = \frac{1+\sqrt{5}}{2}, \gamma = 1$	50.0	9.05	63.2	4	0	0
Choice 2, $\alpha = \frac{1+\sqrt{5}}{2}, \gamma = .9$	51.5	8.91	64.9	6	0	0
Choice 2, $\alpha = \frac{1+\sqrt{5}}{2}, \gamma = .5$	59.4*	7.67*	70.9*	4*	1*	1

TABLE 3.2

Summary test results over the PDE problems. GMLI, GMINS, *and* GMFEE *are geometric means of the numbers of linear iterations, inexact Newton steps, and "function evaluation equivalents," respectively.* NB, NW, *and* NFAIL *are the total numbers of backtracks, instances of convergence to a "wrong" solution, and failures, respectively. Results marked "*" were over successful runs only.*

η_k choice	GMLI	GMINS	GMFEE	NB	NW	NFAIL
10^{-1}	102.2*	11.89*	117.8*	0*	0*	1
10^{-4}	152.4*	6.68*	163.7*	1*	0*	1
$1/2^{k+1}$	104.2*	8.95*	118.4*	3*	0*	1
$\min\{1/(k+2), \Vert F(x_k)\Vert_2\}$	117.6	8.22	130.3	15	1	0
Choice 1	83.5	8.94	96.4	3	0	0
Choice 2, $\alpha = 2, \gamma = 1$	81.7	8.18	93.8	4	0	0
Choice 2, $\alpha = 2, \gamma = .9$	83.3	7.57	95.2	6	0	0
Choice 2, $\alpha = 2, \gamma = .5$	98.4	7.57	110.4	10	0	0
Choice 2, $\alpha = \frac{1+\sqrt{5}}{2}, \gamma = 1$	79.6	8.80	91.9	2	0	0
Choice 2, $\alpha = \frac{1+\sqrt{5}}{2}, \gamma = .9$	83.0	8.70	95.9	4	0	0
Choice 2, $\alpha = \frac{1+\sqrt{5}}{2}, \gamma = .5$	91.9*	6.98*	101.3*	0*	0*	1

equivalents" to be the sum of the numbers of linear iterations, backtracks, and inexact Newton steps. The number of linear iterations is the same as the number of products of F' with vectors; if these products were always approximated by first-order forward differences, then the number of "function evaluation equivalents" would be equal to the number of function evaluations. This number provides a rough relative measure of overall work for the test problems used here. It would be a less suitable measure, e.g., if there were additional costs associated with beginning a new inexact Newton step, such as initializing a new preconditioner. The fourth column gives numbers of backtracks over all test cases, i.e., numbers of step-reductions in the while-loop in Algorithm INB. The fifth column gives numbers of instances of convergence to a "wrong" solution, i.e., convergence to a solution other than the everywhere-positive solution in the PDE problem of §3.2.1 or to a solution other than $u \equiv 1$ in the integral equation problem of §3.2.5. As noted previously, convergence to a "wrong" solution illustrates the potentially serious effects of disagreement between F and its local linear model. The sixth column gives the number of failures over all test cases. If failure occurred in a test case, then that case was not included in the statistics for columns 1–5; consequently, those statistics are not fully comparable to those for which all runs were successful.

One sees from Tables 3.1 and 3.2 that the best overall performances were from Choice 1 and from Choice 2 with $\gamma = .9$ and $\gamma = 1$. Taking $\gamma = .5$ in Choice 2 resulted in significantly less efficiency with $\alpha = 2$; in addition, it led to increased numbers of backtracks with $\alpha = 2$ and to one failure and one instance of convergence to a "wrong" solution with $\alpha = (1+\sqrt{5})/2$, which suggest less robustness when γ is as small as .5. The other choices included in the tests were notably less effective.

Among Choice 1 and Choice 2 with $\gamma = .9$ and $\gamma = 1$, Choice 2 with $\gamma = 1$ and $\alpha = (1 + \sqrt{5})/2$ placed first in every category except mean numbers of inexact Newton steps; thus this choice might be judged the winner. However, its margin of superiority was slight: for example, in "function evaluation equivalents," the best and worst means for these choices differ by less than 4% over all problems and by less than 5% over the PDE problems. Furthermore, there was considerable variance in the relative performance and ranking of these choices among the individual test cases.

The results for Choice 2 illustrate that more aggressive choices of the forcing terms, i.e., choices that are smaller or result in faster asymptotic convergence, may decrease the number of inexact Newton steps up to a point but, through oversolving, may also lead to more linear iterations, more backtracking, and less robustness. Less aggressive choices, on the other hand, may reduce the number of linear iterations up to a point and improve robustness but may also result in increased numbers of inexact Newton steps.

4. Summary discussion.

We have outlined forcing term choices that result in desirably fast local convergence and also tend to avoid oversolving the Newton equation, i.e., imposing an accuracy on an approximation of the Newton step that leads to significant disagreement between F and its local linear model. The choices, along with theoretical support and practical safeguards, are given in §2. Practical performance on a representative set of test problems is discussed in §3.

Choice 1 directly reflects the agreement between F and its local linear model at the previous step. It results in a certain q-superlinear local convergence; see Theorem 2.2 and the following remark for precise statements. Choice 2 does not directly reflect the agreement between F and its local linear model; however, it performed effectively in our tests. Furthermore, it can give up to q-quadratic local convergence (see Theorem 2.3), and the parameters α and γ appearing in it allow flexibility that may be useful in applications.

The best performances in our tests were from Choice 1 and from Choice 2 with $\gamma = .9$ and $\gamma = 1$. (With Choice 2, the values $\alpha = 2$ and $\alpha = (1 + \sqrt{5})/2$ were used in the tests. The latter value was chosen to give convergence roughly comparable to that for Choice 1.) Of these choices, Choice 2 with $\gamma = 1$ and $\alpha = (1 + \sqrt{5})/2$ could be considered most effective in these tests, but only by a small margin; any of these choices might be best for a particular application.

The numerical results in §3 illustrate that, in a globalized Newton iterative or truncated Newton method such as the implementation of Algorithm INB used here, oversolving resulting from inappropriately small forcing terms not only may incur unnecessary expense in solving the Newton equation but also may place significant demands on the globalization and even cause it to fail. In addition, unless special care is taken, very small forcing terms may call for more residual reduction than the iterative linear solver can accurately obtain, especially when finite differences are used to approximate products of F' with vectors. Conversely, choosing larger forcing terms may reduce oversolving and avoid inaccuracy in the iterative linear solver but increase the number of the inexact Newton steps required for convergence.

REFERENCES

[1] P. Bjørstad, *Fast numerical solution of the biharmonic Dirichlet problem on rectangles*, SIAM J. Numer. Anal., 20 (1983), pp. 59–71.

[2] P. N. Brown and Y. Saad, *Hybrid Krylov methods for nonlinear systems of equations*, SIAM J. Sci. Stat. Comput., 11 (1990), pp. 450–481.

[3] X.-C. Cai, W. D. Gropp, D. E. Keyes, and M. D. Tidriri, *Newton–Krylov–Schwarz methods in* CFD, in Proceedings of the International Workshop on the Navier–Stokes Equations, R. Rannacher, ed., Notes in Numerical Fluid Mechanics, Braunschwieg, 1994, Vieweg-Verlag, to appear.

[4] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *Inexact Newton methods*, SIAM J. Numer. Anal., 19 (1982), pp. 400–408.

[5] R. S. Dembo and T. Steihaug, *Truncated Newton algorithms for large-scale optimization*, Math. Programming, 26 (1983), pp. 190–212.

[6] J. E. Dennis, Jr. and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Series in Automatic Computation, Prentice-Hall, Englewood Cliffs, NJ, 1983.

[7] S. C. Eisenstat and H. F. Walker, *Globally convergent inexact Newton methods*, SIAM J. Optimization, 4 (1994), pp. 393–422.

[8] R. Glowinski, H. B. Keller, and L. Reinhart, *Continuation-conjugate gradient methods for the least squares solution of nonlinear boundary value problems*, SIAM J. Sci. Stat. Comput., 6 (1985), pp. 793–832.

[9] C. T. Kelley and J. I. Northrup, *A pointwise quasi-Newton method for integral equations*, SIAM J. Numer. Anal., 25 (1988), pp. 1138–1155.

[10] P. J. McKenna, Private communication, 1992.

[11] J. J. Moré, *A collection of nonlinear model problems*, in Computational Solution of Nonlinear Systems of Equations, E. L. Allgower and K. Georg, eds., Lectures in Applied Mathematics Vol. 26, American Mathematical Society, Providence, RI, 1990, pp. 723–762.

[12] Y. Saad and M. H. Schultz, GMRES: *A generalized minimal residual method for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.

[13] R. Schaaf, Private communication, 1994.

[14] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, Springer-Verlag, New York, 1980.

[15] P. N. Swartztrauber and R. A. Sweet, *Algorithm* 541: *Efficient Fortran subprograms for the solution of separable elliptic partial differential equations*, ACM Trans. Math. Software, 5 (1979), pp. 352–364.

[16] K. Turner and H. F. Walker, *Efficient high accuracy solutions with* GMRES(m), SIAM J. Sci. Stat. Comput., 13 (1992), pp. 815–825.

[17] C. J. van Duijn and J. M. de Graaf, *Large time behaviour of solutions of the porous medium equation with convection*, J. Differential Equations, 84 (1990), pp. 183–203.

SIAM J. SCI. COMPUT.
Vol. 17, No. 1, pp. 33–46, January 1996

FAST NONSYMMETRIC ITERATIONS AND PRECONDITIONING FOR NAVIER–STOKES EQUATIONS*

HOWARD ELMAN[†] AND DAVID SILVESTER[‡]

Abstract. Discretization and linearization of the steady-state Navier–Stokes equations gives rise to a nonsymmetric indefinite linear system of equations. In this paper, we introduce preconditioning techniques for such systems with the property that the eigenvalues of the preconditioned matrices are bounded independently of the mesh size used in the discretization. We confirm and supplement these analytic results with a series of numerical experiments indicating that Krylov subspace iterative methods for nonsymmetric systems display rates of convergence that are independent of the mesh parameter. In addition, we show that preconditioning costs can be kept small by using iterative methods for some intermediate steps performed by the preconditioner.

Key words. Navier–Stokes, iterative methods, preconditioners, Krylov subspace

AMS subject classifications. 65F10, 65N12, 65N22, 65M60

1. Introduction.

Consider the steady-state *Navier–Stokes problem*: given data $\mathbf{f}$, find the velocity $\mathbf{u}$ and pressure p satisfying

$$\begin{aligned} -\nu\nabla^2\mathbf{u} + \frac{1}{2}\mathbf{u}(\operatorname{div}\mathbf{u}) + \mathbf{u}\cdot\nabla\mathbf{u} + \operatorname{grad} p &= \mathbf{f} \qquad \text{in } \Omega \\ \operatorname{div}\mathbf{u} &= 0 \end{aligned} \tag{1.1}$$

subject to boundary conditions, typically, specified velocity $\mathbf{u} = \mathbf{g}$ on $\partial\Omega$; $\Omega \subset \mathbf{R}^2$ or $\Omega \subset \mathbf{R}^3$. Here, the scalar ν is the inverse of the Reynolds number or the ratio of convection to diffusion in the system. In the diffusion dominated case, ($\nu \to \infty$) (1.1) tends to a linear self-adjoint system of equations known as the *Stokes problem.*

There are two ways of calculating solutions to the system (1.1). A popular approach is to compute "true" steady-state solutions of the time-dependent Navier–Stokes equations. There are many ways to do this: one way is to make use of the "characteristics" associated with the hyperbolic part of the Navier–Stokes operator via a Lagrange–Galerkin approach (for example, see [13]). The associated transpose-diffusion splitting leads to absolutely stable temporal discretizations so that large time steps can be taken. At each time step, a symmetric indefinite matrix system corresponding to a time-discretized Stokes-like system must be solved. These systems can be solved efficiently by iterative methods, for example, if a multigrid solver is used to precondition the primary (Laplacian) operator. There are, however, a number of disadvantages to the time-dependent approach. Simple time discretization methods based on the l_2-projection onto the discretely divergence-free subspace [10] have an $O(h)$ CFL restriction on the time step, which impinges on efficiency. On the other hand, absolutely stable schemes like the method of backward characteristics are known to be sensitive to implementation issues (e.g., the need to perform quadrature; see [13]). Even with fixed grids, efficiency is often limited by the costs associated with interpolation.

In this work, we consider the alternative approach of attacking the system (1.1) directly. Applying a fixed point (or Picard) iteration, the system (1.1) reduces to solving a sequence of

*Received by the editors June 8, 1994; accepted for publication (in revised form) January 21, 1995. This work was supported by the U. S. Army Research Office under grant DAAL-0392-G-0016 and the U. S. National Science Foundation under grant ASC-8958544 at the University of Maryland and the Science and Engineering Research Council of Great Britain Visiting Fellowship GR/J 04982 at the University of Manchester Institute of Science and Technology.

[†] Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742 (elman@cs.umd.edu).

[‡] Department of Mathematics, University of Manchester Institute of Science and Technology, Manchester M601QD, UK (na.silvester@na-net.ornl.gov).

linear Oseen problems of the following form: given some velocity field $\mathbf{w}$, find the velocity $\mathbf{u}$ and pressure p satisfying

$$-\nu\nabla^2\mathbf{u} + \frac{1}{2}\mathbf{u}(\operatorname{div}\mathbf{w}) + \mathbf{w}\cdot\nabla\mathbf{u} + \operatorname{grad} p = \mathbf{f} \qquad \text{in } \Omega \tag{1.2}$$
$$\operatorname{div}\mathbf{u} = 0$$

subject to the same boundary conditions.

For this methodology to be effective it is necessary to solve the discrete versions of (1.2) efficiently. Finite element discretization of (1.2) leads to the matrix problem

$$\begin{pmatrix} \nu A + N & B^t \\ B & O \end{pmatrix}\begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}, \tag{1.3}$$

where $A = A^t$ represents diffusion (for example, $-\nabla^2$) and hence is a positive definite matrix of order n_u; N represents convection ($\frac{1}{2}\operatorname{div}\mathbf{w} + \mathbf{w}\cdot\nabla$); and the $n_p \times n_u$ matrix B represents the coupling between the discrete velocity u and the pressure p. Note that the representation of the quadratic convection term in (1.1) ensures that if velocity is specified everywhere on the boundary then $N = -N^t$; that is, the discrete form of the convection operator is *skew-symmetric* [10, p. 53]. Note that with these boundary conditions the system (1.3) is singular; pressure is only unique up to a (hydrostatic) constant. We assume in the following analysis that the pressure solution is uniquely specified in this case, e.g., by insisting that its mean is zero.

Working in a conventional mixed finite element framework, we will further assume that the underlying velocity and pressure approximations are *(div-)stable* (see, e.g., [2, p. 57], [10, pp. 10ff], and [19]); i.e., defining a mesh parameter h, a velocity space $\mathbf{V}_h$, and a pressure space P_h, there exist constants γ, Γ, independent of h, such that

$$\gamma^2 \le \frac{(p, BA^{-1}B^t p)}{(p, Qp)} \le \Gamma^2 \qquad \forall p \in P_h\,. \tag{1.4}$$

Here, Q is the pressure mass matrix, or alternatively the Grammian matrix of basis functions defining P_h. The lower bound γ is the so-called *inf–sup* constant. The relation (1.4) is crucial to the success of iterative solvers for solving discrete Stokes problems because it implies that, using a quasi-uniform mesh, the Schur complement $BA^{-1}B^t$ has a condition number bounded independently of h. It is also known from our previous work [16] that when $\nu \to \infty$, "optimal" preconditioners for the Laplacian subblocks give rise to "optimal" preconditioners for the Stokes problem in the sense that the spectra of the underlying discrete operators are contained in small clusters, which are bounded independently of h. A consequence of this is that the asymptotic convergence rate, with respect to the preconditioned residual norm, of Krylov subspace methods applied to discrete Stokes problems is also independent of h.

In this paper, we derive analogous results for eigenvalues in the general Oseen case. We introduce two preconditioners for the Oseen problem such that, for any value $0 < \nu < \infty$, the eigenvalues of the preconditioned Oseen operator are bounded independently of the mesh size. These observations apply to arbitrary discretizations satisfying (1.4). In addition, in a series of numerical experiments we show that these bounds on eigenvalues are predictive of the performance of Krylov subspace iterative methods for solving the preconditioned Oseen equations. Of course, it is well known that when convection dominates (i.e., when ν is "small" relative to h and $\|\mathbf{w}\|$), the standard Galerkin approximation deteriorates. Oscillations in the discrete velocity are apparent if the local mesh Reynolds number $Re^h = h\|\mathbf{w}\|/\nu$ is greater than unity. In such situations, the addition of streamwise diffusion to the discrete system is known to give added stability, both theoretically and numerically; see [3] and [12]. In our experiments, we demonstrate the effectiveness of the ideas using both a standard Galerkin discretization on a set of quasi-uniform grids and a streamline-upwind scheme on a set of uniform grids.

The remainder of the paper is divided into three sections. Our main theoretical results are presented in §2, and results of numerical experiments confirming and augmenting the theoretical analysis are given in §3. In §4, we consider more practical preconditioning strategies and present a perturbation analysis and additional numerical experiments demonstrating their effectiveness.

2. Preconditioning strategies. In this section, we introduce two preconditioning techniques for (1.1) and present an analysis showing that the spectra of the preconditioned systems are bounded independently of the discretization mesh size h. Throughout the section, we will be concerned with the eigenvalues of preconditioned matrices; these matrices can be viewed as being of the form $\mathcal{A}\mathcal{M}^{-1}$, where $\mathcal{A}$ is the original matrix and $\mathcal{M}$ is the preconditioner. Equivalently, we are concerned with the solution of the generalized eigenvalue problem $\mathcal{A}v = \lambda\mathcal{M}v$. All the matrices in question are implicitly parameterized by h. For simplicity, we state our results under the assumption that B of (1.3) has full rank.

The first idea is derived from a method developed in [14], [16], [20] for the discrete Stokes equations, where the coefficient matrix has the form

$$\begin{pmatrix} A & B^t \\ B & O \end{pmatrix}. \tag{2.1}$$

Consider the preconditioner

$$\begin{pmatrix} A & 0 \\ 0 & Q \end{pmatrix}$$

for (2.1). The eigenvalues of the preconditioned operator are then given by the solution to the generalized eigenvalue problem

$$\begin{pmatrix} A & B^t \\ B & O \end{pmatrix}\begin{pmatrix} u \\ p \end{pmatrix} = \lambda \begin{pmatrix} A & 0 \\ 0 & Q \end{pmatrix}\begin{pmatrix} u \\ p \end{pmatrix}. \tag{2.2}$$

One solution is $\lambda = 1$, of multiplicity $n_u - n_p$, for which the eigenvectors have the form $\binom{u}{0}$ where $Bu = 0$; i.e., u is "discretely divergence free." The remaining eigenvalues come from the solution of the quadratic equation $\lambda(\lambda - 1) = \mu$, where μ is a generalized eigenvalue of the Schur complement associated with (2.1),

$$BA^{-1}B^t p = \mu Qp. \tag{2.3}$$

Equivalently,

$$\lambda = \frac{1 \pm \sqrt{1+4\mu}}{2}. \tag{2.4}$$

Since (1.4) implies that as $h \to 0$ the solutions to (2.3) remain bounded above and below, it follows that the eigenvalues of (2.2) are also bounded. The preconditioned conjugate residual method can then be used to solve (2.1), with a convergence rate independent of h [14], [16].

A natural generalization for the discrete Oseen equations uses the block preconditioner

$$\begin{pmatrix} F & 0 \\ 0 & \frac{1}{\nu}Q \end{pmatrix}, \tag{2.5}$$

where $F = \nu A + N$. As above, the generalized eigenvalues for

$$\begin{pmatrix} F & B^t \\ B & O \end{pmatrix}\begin{pmatrix} u \\ p \end{pmatrix} = \lambda \begin{pmatrix} F & 0 \\ 0 & \frac{1}{\nu}Q \end{pmatrix}\begin{pmatrix} u \\ p \end{pmatrix} \tag{2.6}$$

are either $\lambda = 1$ or (2.4), where μ is now a solution to the generalized eigenvalue problem

$$Sp = \mu\left(\frac{1}{\nu}Q\right)p, \tag{2.7}$$

with $S = BF^{-1}B^t$, the Schur complement for the discrete Oseen operator. The following result, which generalizes the analysis for the Stokes operator in [19], provides a bound.

THEOREM 1. *The eigenvalues of the generalized Schur complement problem* (2.7) *for the Oseen operator are contained in a rectangular box in the right half plane whose borders are bounded independently of h.*

Proof. Let $C = B\left(\frac{F^{-1}+F^{-t}}{2}\right)B^t$ denote the symmetric part of S, and let $R = B\left(\frac{F^{-1}-F^{-t}}{2}\right)B^t$ denote its skew-symmetric part, so that $S = C + R$. By Bendixson's theorem ([17, p. 418]), any eigenvalue μ of the problem (2.7) satisfies

$$\min_p \frac{(p, Cp)}{(p, \frac{1}{\nu}Qp)} \le \mathrm{Re}(\mu) \le \max_p \frac{(p, Cp)}{(p, \frac{1}{\nu}Qp)}, \qquad |\mathrm{Im}(\mu)| \le \max_p \frac{|(p, Rp)|}{(p, \frac{1}{\nu}Qp)}. \tag{2.8}$$

To construct bounds on these Rayleigh quotients, it will be convenient to refer to $S_\infty = BA^{-1}B^t$, the Schur complement for the Stokes operator. For the symmetric part C in (2.8), we use the relation

$$\frac{(p, Cp)}{(p, \frac{1}{\nu}Qp)} = \frac{(p, Cp)}{(p, \frac{1}{\nu}S_\infty p)}\,\frac{(p, S_\infty p)}{(p, Qp)}. \tag{2.9}$$

In light of (1.4), we need only consider the first quotient on the right in (2.9). Note that

$$\begin{aligned} \frac{F^{-1}+F^{-t}}{2} &= F^{-1}\left(\frac{F+F^t}{2}\right)F^{-t} \\ &= (\nu A + N)^{-1}(\nu A)(\nu A - N)^{-1} \\ &= A^{-1/2}\left(\nu I - \frac{1}{\nu}\tilde{N}^2\right)^{-1} A^{-1/2}, \end{aligned} \tag{2.10}$$

where $\tilde{N} = A^{-1/2}NA^{-1/2}$. Consequently,

$$\frac{(p, Cp)}{(p, \frac{1}{\nu}S_\infty p)} = \frac{(p, BA^{-1/2}(\nu I - \frac{1}{\nu}\tilde{N}^2)^{-1}A^{-1/2}B^t p)}{(p, \frac{1}{\nu}BA^{-1}B^t p)} = \frac{(v, (I - \frac{1}{\nu^2}\tilde{N}^2)^{-1}v)}{(v, v)},$$

where $v = A^{-1/2}B^t p$. But $\tilde{N}$ is skew-symmetric, so that the eigenvalues of $-\tilde{N}^2$ are real and nonnegative. Moreover, since N and A are first-order and second-order operators, respectively, the eigenvalues of $\tilde{N}$ are uniformly bounded in modulus by a constant δ that is independent of h [5]. Therefore, the spectrum of $I - \frac{1}{\nu^2}\tilde{N}^2$ is contained in the interval $\left[1, 1 + \delta^2/\nu^2\right]$, or, equivalently,

$$\frac{\nu^2}{\delta^2 + \nu^2} \le \frac{(p, Cp)}{(p, \frac{1}{\nu}S_\infty p)} \le 1.$$

Combining this with (1.4) and (2.9) gives

$$\frac{\gamma^2\nu^2}{\delta^2 + \nu^2} \le \frac{(p, Cp)}{(p, \frac{1}{\nu}Qp)} \le \Gamma^2.$$

For the skew-symmetric part R in (2.8), the analogue of (2.9) is

$$\frac{(p, Rp)}{(p, \frac{1}{\nu}Qp)} = \frac{(p, Rp)}{(p, \frac{1}{\nu}S_\infty p)} \, \frac{(p, S_\infty p)}{(p, Qp)} \, ,$$

and as in (2.10), we have

$$\frac{F^{-1} - F^{-t}}{2} = -A^{-1/2} \, (\nu I + \tilde{N})^{-1} \, \tilde{N} \, (\nu I - \tilde{N})^{-1} \, A^{-1/2}.$$

Therefore

$$\frac{(p, Rp)}{(p, \frac{1}{\nu}S_\infty p)} = -\frac{\nu(v, \tilde{N}v)}{(v, (\nu^2 I - \tilde{N}^2)v)} \, , \tag{2.11}$$

where $v = (\nu I - \tilde{N})^{-1} A^{-1/2} B^T p$. The skew-symmetric matrix $\tilde{N}$ admits a decomposition of the form $\tilde{N} = iU\Lambda U^H$, where Λ is a real diagonal matrix and U is unitary. Consequently, $\tilde{N}^2 = -U\Lambda^2 U^H$, and the modulus of the Rayleigh quotient on the right side of (2.11) can be expressed in the form

$$\frac{\nu \, |(w, \Lambda w)|}{(w, (\nu^2 I + \Lambda^2)w)} \, .$$

This is bounded by

$$\max_{-\delta \le \lambda \le \delta} \frac{\nu|\lambda|}{\nu^2 + \lambda^2} = \max_{0 \le \lambda \le \delta} \frac{\nu\lambda}{\nu^2 + \lambda^2} \, .$$

It follows from elementary calculus that this maximum is $\frac{1}{2}$, obtained when $\lambda = \nu$, giving

$$\frac{|(p, Rp)|}{(p, \frac{1}{\nu}Qp)} \le \frac{\Gamma^2}{2} \, . \qquad \square$$

Corollary 1 follows immediately from Theorem 1 and (2.4).

COROLLARY 1. *The eigenvalues of the discrete Oseen operator preconditioned by* (2.5) *consist of* $\lambda = 1$ *of multiplicity* $n_u - n_p$, *together with four sets consisting of points of the form* $1 + (a \pm bi)$ *and* $-a \pm bi$. *These sets can be enclosed in two rectangular regions that are symmetric with respect to* $\mathrm{Re}\,(\lambda) = \frac{1}{2}$, *whose borders are bounded independently of* h.

The inclusion regions for these eigenvalues consist of the image of the box $\left[\frac{\gamma^2\nu^2}{\delta^2+\nu^2}, \Gamma^2\right] \times \left[-\frac{\Gamma^2}{2}, \frac{\Gamma^2}{2}\right]$ under the mapping $\mu \mapsto \lambda(\mu)$ given by (2.4). It can be shown that the rectangular regions of this result are contained in

$$\left[\frac{1 + s_{\min}}{2}, \frac{1 + s_{\max}}{2}\right] \times [-t, t] \quad \text{and} \quad \left[\frac{1 - s_{\max}}{2}, \frac{1 - s_{\min}}{2}\right] \times [-t, t]$$

in the right and left half sides, respectively, of the complex plane, where

$$s_{\min} = \left(1 + \frac{4\gamma^2\nu^2}{\delta^2 + \nu^2}\right)^{1/2}, \qquad s_{\max} = \left[\frac{1}{2}\left(1 + 4\Gamma^2 + \sqrt{1 + 8\gamma^2 + 20}\,\right)\right]^{1/2},$$

$$t = \frac{\Gamma^2}{\left(1 + \frac{4\gamma^2\nu^2}{\delta^2+\nu^2}\right)^{1/2}} \, .$$

The fact that the eigenvalues for the preconditioned system derived from (2.5) lie on both sides of the imaginary axis is a potential disadvantage of this idea. An alternative that avoids this problem is the block triangular operator

$$\begin{pmatrix} F & B^t \\ 0 & -\frac{1}{\nu}Q \end{pmatrix}. \tag{2.12}$$

For this choice, the preconditioned eigenvalue problem is

$$\begin{pmatrix} F & B^t \\ B & O \end{pmatrix}\begin{pmatrix} u \\ p \end{pmatrix} = \lambda \begin{pmatrix} F & B^t \\ 0 & -\frac{1}{\nu}Q \end{pmatrix}\begin{pmatrix} u \\ p \end{pmatrix}. \tag{2.13}$$

Again, one solution is $\lambda = 1$, now of multiplicity n_u. If $\lambda \neq 1$, then premultiplying the first block row of (2.13) by BF^{-1} and using the relation $Bu = -\lambda(\frac{1}{\nu}Q)p$ leads to equation (2.7) for the other eigenvalues. Thus, we have the following result.

THEOREM 2. *The eigenvalues of the discrete Oseen operator preconditioned by* (2.12) *consist of* $\lambda = 1$ *together with the generalized eigenvalues of* S *in* (2.7). *Therefore, the eigenvalues are bounded independently of* h.

Remark 1. Use of either preconditioning operator (2.5) or (2.12) entails the computation of the action of F^{-1} at each step of an iterative procedure. F is a discrete convection-diffusion operator and applying F^{-1} to a vector using direct methods will be expensive. An alternative is to replace this computation with an approximation obtained by iterative solution of the convection-diffusion equation. We will examine this approach in §4.

Remark 2. Both preconditioners also require the action of Q^{-1}, which may also be expensive, depending on the choice of pressure discretization. In this case, however, it is known that Q can be replaced by some approximation $\hat{Q}$ without affecting asymptotic convergence properties; only the constants γ and Γ of (1.4) change [21]. In the experiments discussed in §§3 and 4, we replace Q with a diagonal matrix consisting of the main diagonal of Q.

3. Numerical results I: Exact convection-diffusion solves. In this section, we present the results of numerical experiments indicating that the analysis of §2 is predictive of the performance of iterative methods for solving (1.3). Computations were performed using MATLAB 4.1 on a variety of computing platforms.

Our test problem is a "leaky" two-dimensional lid-driven cavity problem in a square domain $(-1 \leq x \leq 1 : -1 \leq y \leq 1)$. The boundary conditions are $u_x = u_y = 0$ on the three fixed walls $(x = -1, y = -1, x = 1)$, and $u_x = 1, u_y = 0$ on the moving wall $(y = 1)$. The hydrostatic pressure is not explicitly specified, so that all the linear equation systems we solve below are singular with a one-dimensional nullspace. The convective "wind" is a circular vortex as illustrated in Fig. 1, and is given by

$$\begin{aligned} w_x &= 2y(1 - x^2), \\ w_y &= -2x(1 - y^2). \end{aligned}$$

The fact that there is no dominant flow direction makes this a challenging test problem. Note that in the corners and in the center of the flow region the driving flow is stagnant.[1]

Unless otherwise specified, we consider values of the viscosity parameter ν between 1 and 1/100. When $\nu = 1$ we have diffusion dominated (essentially Stokes) flow, whereas as

[1] This is actually not a linearization of the Navier–Stokes equations since $\mathbf{w}$ and $\mathbf{u}$ do not satisfy the same boundary conditions; this fact does not affect the demonstration of the solution algorithms. Also, note that $\operatorname{div}\mathbf{w} = 0$ so that the term $\frac{1}{2}\mathbf{u}(\operatorname{div}\mathbf{w})$ in (1.2) is identically zero.

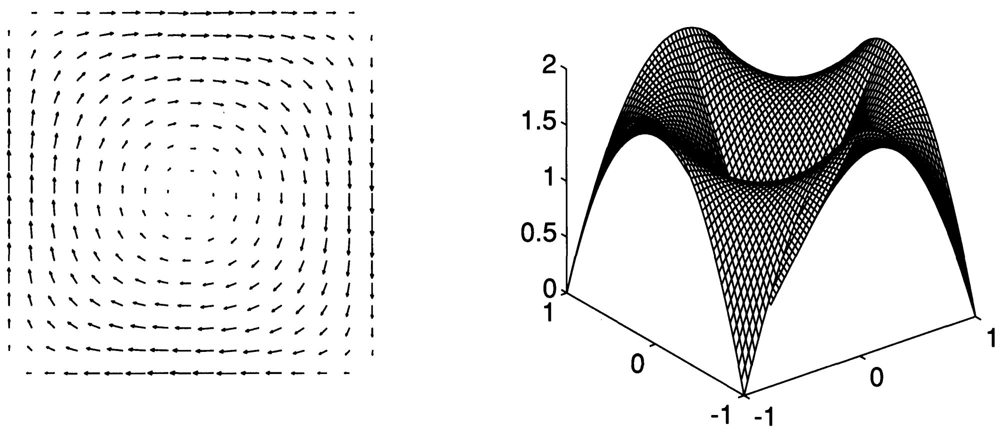


FIG. 1. *Magnitude and direction of the convecting flow.*

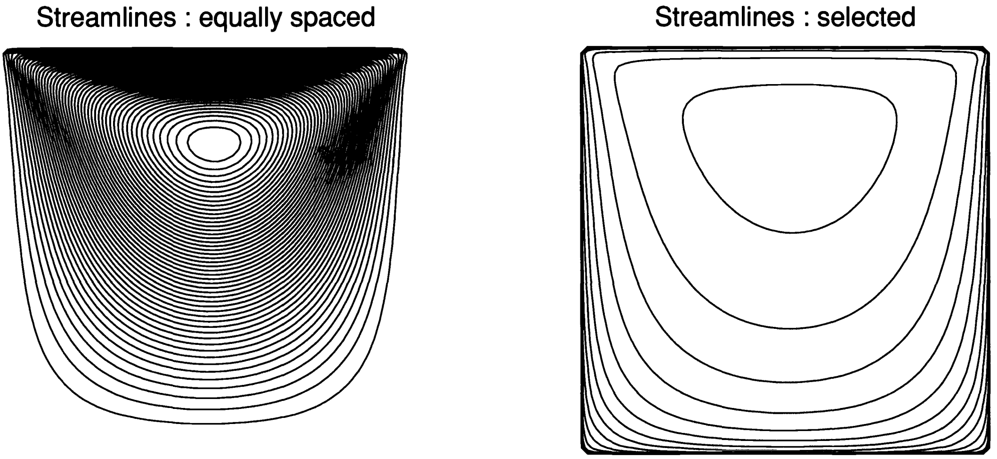


FIG. 2. *Uniform* 64 × 64 *grid*: $\nu = 1$.

$\nu \to 0$ the flow becomes dominated by the "wind." Typical flow solutions are illustrated in Figs. 2 and 3. Note that as the viscosity is decreased, the center of primary recirculation moves to the right (the Stokes flow solution is perfectly symmetric about the line $x = 0$), and secondary vortices are generated in the two bottom corners.

To discretize (1.2), we take a finite element subdivision based on $n \times n$ grids of rectangular elements. Bearing in mind the nature of the flow solution being computed, we present results for two representative discretizations here: a conventional Galerkin approach using a quasi-uniform sequence of grids and a streamline-upwind method using uniform grids of square elements of size $h = 2/n$. In either case, the mixed finite element used was the div-stable "Taylor–Hood" method based on continuous bilinear pressure with a continuous bilinear velocity field defined on four element macroelements (see, e.g., [10, p. 30]).

For the Galerkin discretization, the quasi-uniform grids are chosen to resolve the details of the flow in the four corners of the domain: they are symmetric about $x = 0$ and $y = 0$, and in each quadrant the grid lines become more dense near the boundary. The 64 × 64 grid is shown in the pressure solution plot in Fig. 4. The analytic pressure solution is singular at the top corners where the imposed velocity is discontinuous.

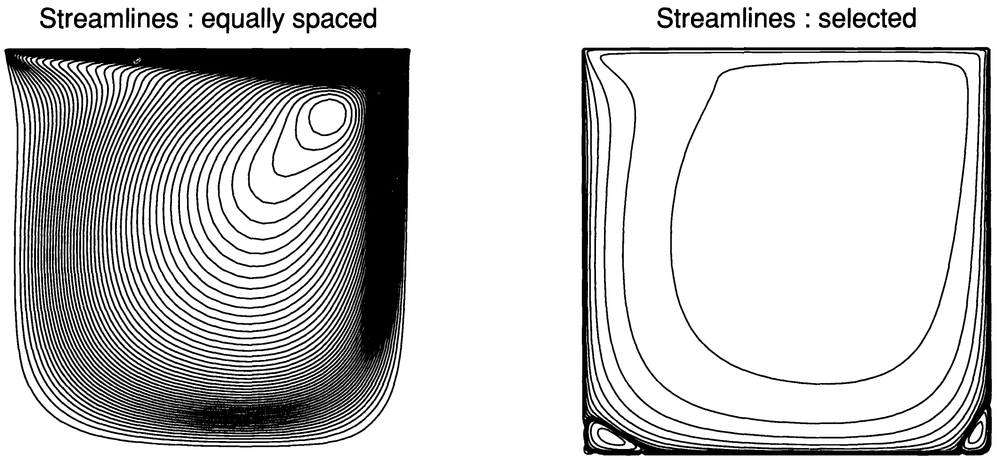


FIG. 3. *Nonuniform* 64 × 64 *grid*: $\nu = 1/100$.

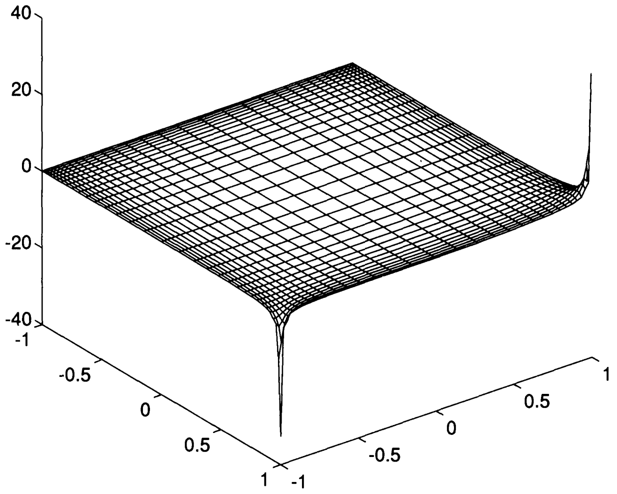


FIG. 4. *Pressure solution for* $\nu = 1/10$.

The streamline-upwind discretization is as described in [12, p. 185]. In this case, the block convection-diffusion operator F is perturbed by a symmetric positive semidefinite matrix $A_{\mathbf{w}}$. That is, $F = -\nu\Delta_h + A_{\mathbf{w}} + N$, where Δ_h is the discrete Laplacian obtained from the usual Galerkin formulation. $A_{\mathbf{w}}$ is the discrete form of a stabilizing term $\alpha\ (\mathbf{w}\cdot\nabla\mathbf{u}, \mathbf{w}\cdot\nabla\mathbf{v})$ that adds $\alpha \equiv O(h)$ diffusion along the streamlines. For our experiments with streamline upwinding, we took $\alpha = h/4$. Note that the perturbation does not affect the skew-symmetric part of the convection-diffusion operator, so that the analysis of §2 holds; only the definition of the "diffusion matrix" A is changed, from $-\Delta_h$ to $-\Delta_h + \frac{1}{\nu}A_{\mathbf{w}}$.

We first consider the bounds of Theorem 1. Table 1 shows the extreme real parts and maximum imaginary parts of the generalized eigenvalues (2.7) of the Schur complement operator, for $\nu = 1/10$ and $1/100$ with the streamline-upwind discretization, on three meshes. The small changes in all values are in accordance with the analysis, although it appears that finer meshes would be needed to produce constant values. The analysis also shows that the

TABLE 1
Eigenvalues of the Schur complement for streamline-upwind discretization.

	$\nu = 1/10$			$\nu = 1/100$		
Grid	Min Re	Max Re	Max Im	Min Re	Max Re	Max Im
16×16	7.17E-2	1.11	0.46	1.66E-2	1.07	0.20
32×32	8.75E-2	1.64	0.71	1.33E-2	1.11	0.50
64×64	9.08E-2	2.00	0.87	1.14E-2	1.37	0.74

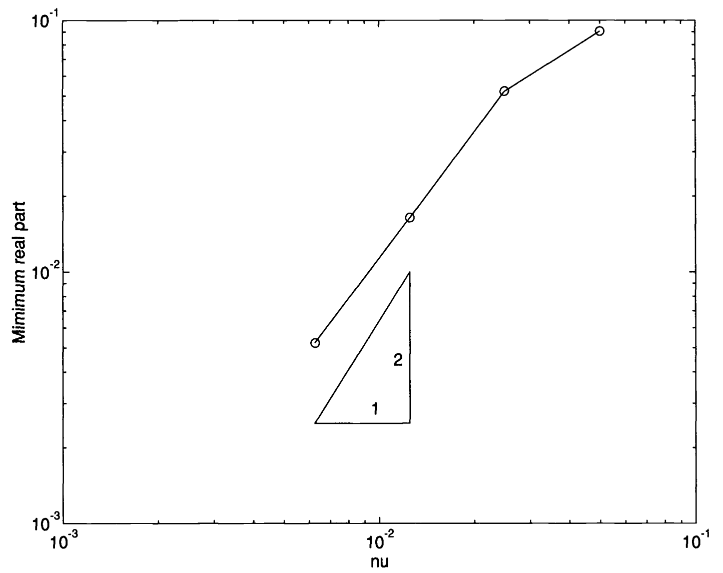


FIG. 5. *Minimum real parts of eigenvalues of the Schur complement, for streamline-upwind discretization on a* 64×64 *grid.*

real parts and largest imaginary parts of the eigenvalues are bounded independently of ν; the bound for the smallest real part is proportional to ν^2. The data in Table 1 are in agreement with the upper bounds. Figure 5 plots the smallest real parts on a logarithmic scale for the streamline-upwind discretization on a 64×64 grid and $\nu = 1/20$, $1/40$, $1/80$, and $1/160$. The results indicate that the lower bound is also tight.

We test the preconditioners here with two Krylov subspace methods for solving nonsymmetric systems: the generalized minimum residual method (GMRES) [15] and a simple implementation of the quasi-minimum residual method (QMR) [8] based on coupled two-term recurrences without look-ahead. GMRES demonstrates the performance of the preconditioners with the optimal (with respect to the residual norm) Krylov subspace solver. This method is impractical for large problems because its work and storage requirements grow with the iteration count; QMR is a nonoptimal alternative that avoids this difficulty. Some additional experiments with restarted GMRES are presented in §4. In all cases we use right-oriented preconditioning, and our convergence criterion is a reduction of 10^{-6} in the l_2-norm of the residual. The action of F^{-1} and F^{-t} is computed using the LU-factorization in MATLAB. We start from a zero initial guess. Using random initial guesses gave comparable iteration counts in all cases.

We first discuss the performance of GMRES. Table 2 shows the iteration counts obtained for three values of ν using the block triangular preconditioner (2.12) in the case of Galerkin approximation on the quasi-uniform grid sequence, and Table 3 shows the iteration counts for uni-

TABLE 2
GMRES iteration counts for Galerkin discretization with block triangular preconditioner.

Grid	16 x 16	32 x 32	64 x 64	128 x 128
$\nu = 1$	18	19	17	14
$\nu = 1/10$	25	31	32	31
$\nu = 1/50$	45	69	93	110

TABLE 3
GMRES iteration counts for streamline-upwind discretization with block triangular preconditioner.

Grid	16 x 16	32 x 32	64 x 64	128 x 128
$\nu = 1$	21	22	21	19
$\nu = 1/10$	32	36	35	33
$\nu = 1/50$	48	72	97	111

TABLE 4
QMR iteration counts for Galerkin discretization with block triangular (diagonal) preconditioner.

Grid	16 x 16	32 x 32	64 x 64	128 x 128
$\nu = 1$	22 (43)	22 (43)	22 (41)	16
$\nu = 1/10$	28 (51)	36 (68)	39 (78)	36
$\nu = 1/50$	51 (100)	78 (155)	112 (223)	127
$\nu = 1/100$	73 (143)	126 (246)	189 (375)	253

form grids with streamline-upwind discretization. The results suggest that grid-independent convergence is observed even on relatively coarse grids if the flow is diffusion dominated, that is, eigenvalues are indeed predictive of performance. If convection dominates (as ν tends to zero), then the iteration counts increase, as might be expected from the analytic bounds of §2. For the smallest value of ν considered here, $1/50$, the iteration counts grow as the mesh is refined; although for fine enough grids the counts become close to constant. We believe that for "small" ν, the asymptotic grid independence will be visible only for sufficiently fine grids.

GMRES with the block diagonal preconditioner (2.5) gives an identical picture. Indeed, we find that for the grid sizes and values of ν in Tables 2 and 3 (only $n \leq 64$ was tested), GMRES requires precisely $2k - 1$ iterations to reach the tolerance where k is the iteration count from Table 2 or 3. Moreover, the behavior of GMRES using (2.5) mirrors its behavior with the triangular preconditioning, in the sense that the odd iterates at step $2i - 1$ are close to those obtained in the triangular case at step i, and the norms of the even iterates stagnate. A rigorous explanation for this behavior can be given by relating the optimal polynomials implicitly generated by GMRES. In particular, it follows from the analysis in [6] that for a particular starting guess (not zero), the ith GMRES polynomial for the triangular case is identical to the $(2i - 1)$st GMRES polynomial for the diagonal case.

Tables 4 and 5 show analogous iteration counts for QMR. The tables contain data for both the triangular and (in parentheses for $n \leq 64$) diagonal preconditioners. Note that the iteration counts in the first three rows of Tables 4 and 5 are close to the corresponding entries of Tables 2 and 3, respectively; i.e., the performance in QMR is close to optimal. Results for $\nu = 1/100$ are included to get a sense of the behavior of the preconditioners as ν becomes small; it appears that the asymptotic behavior is not seen for the grid sizes used.

Note that the cost per step of the block triangular preconditioner is only slightly higher than that of the block diagonal preconditioner (only an extra multiplication by B^t is needed); hence (2.12) is more efficient. However, for the Stokes problem ($\nu \to \infty$), the preferred choice of preconditioner is less obvious since the inherent symmetry is destroyed if (2.12) is used in place of (2.5).

TABLE 5

QMR iteration counts for streamline-upwind discretization with block triangular (diagonal) preconditioner.

Grid	16 x 16	32 x 32	64 x 64	128 x 128
$\nu = 1$	25 (49)	27 (51)	25 (47)	22
$\nu = 1/10$	36 (78)	44 (91)	42 (80)	37
$\nu = 1/50$	61 (127)	91 (175)	117 (234)	130
$\nu = 1/100$	76 (157)	133 (249)	190 (382)	229

Remark 3. In addition to the implementation of QMR with a coupled two-term recurrence (QMR$_2$) discussed above, we tested a version without look-ahead based on a three-term recurrence (QMR$_3$) [7], and the definitive (Fortran) implementation of a two-term QMR with look-ahead (QMR$_2^*$) from the QMRPAK directory in Netlib. For the preconditioners discussed above, the performances of the three variants were virtually identical. However, with the inexact preconditioners of the next section, we found QMR$_2$ to be much more robust than QMR$_3$.

4. Numerical results II: Inexact convection-diffusion solves. The dominant costs of the preconditioners of §§2 and 3 come from applying the action of F^{-1}, and for QMR, F^{-t}, to some vector v at each step of the iteration. In this section, we show that this operation can be replaced by an inexpensive one derived from an approximation to F^{-1}, with little degradation of performance of the Krylov subspace methods. The idea is to replace the preconditioning operators (2.5) and (2.12) with

$$(4.1) \qquad \begin{pmatrix} \hat{F} & 0 \\ 0 & \frac{1}{\nu}Q \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} \hat{F} & B^t \\ 0 & -\frac{1}{\nu}Q \end{pmatrix},$$

respectively, where $\hat{F} \approx F$. Our choice of $\hat{F}$ will be implicitly determined by the use of iterative methods to compute approximate solutions to the systems $Fw = v$ and $F^t w = v$, although the methodology is not restricted to this choice. We will refer to the preconditioners that use the exact action of F^{-1} as the *exact* versions, and those based on approximations as in (4.1) as *inexact* versions.

Some insight into the effects of the inexact preconditioners is derived from matrix perturbation theory. Let $Q_\nu = \frac{1}{\nu}Q$. The preconditioned matrix for the exact block diagonal preconditioner (2.5) is

$$\mathcal{A}_D = \begin{pmatrix} F & B^t \\ B & O \end{pmatrix} \begin{pmatrix} F & 0 \\ 0 & Q_\nu \end{pmatrix}^{-1} = \begin{pmatrix} I & B^t Q_\nu^{-1} \\ BF^{-1} & 0 \end{pmatrix},$$

and that derived from the inexact version is

$$\hat{\mathcal{A}}_D = \begin{pmatrix} F & B^t \\ B & O \end{pmatrix} \begin{pmatrix} \hat{F} & 0 \\ 0 & Q_\nu \end{pmatrix}^{-1} = \mathcal{A}_D + \mathcal{E}_D,$$

where, with $E = \hat{F} - F$,

$$\mathcal{E}_D = -\begin{pmatrix} E\hat{F}^{-1} & 0 \\ BF^{-1}E\hat{F}^{-1} & 0 \end{pmatrix}.$$

Similarly, the preconditioned matrices for the exact and inexact block tridiagonal preconditioners (2.12) satisfy $\hat{\mathcal{A}}_T = \mathcal{A}_T + \mathcal{E}_T$, where

$$\mathcal{A}_T = \begin{pmatrix} I & 0 \\ BF^{-1} & BF^{-1}B^t Q_\nu^{-1} \end{pmatrix}, \qquad \mathcal{E}_T = -\begin{pmatrix} E\hat{F}^{-1} & E\hat{F}^{-1}B^t Q_\nu^{-1} \\ BF^{-1}E\hat{F}^{-1} & BF^{-1}E\hat{F}^{-1}B^t Q_\nu^{-1} \end{pmatrix}.$$

We have the following bounds on the eigenvalues of the preconditioned systems using inexact preconditioners.

THEOREM 3. *If* $\mathcal{A}_D = \mathcal{V}_D \Lambda_D \mathcal{V}_D^{-1}$ *is diagonalizable, then for any eigenvalue* $\mu \in \sigma(\hat{\mathcal{A}}_D)$,

$$\min_{\lambda \in \sigma(\mathcal{A}_D)} |\lambda - \mu| \leq \|E\hat{F}^{-1}\|_\infty \kappa_\infty(\mathcal{V}_D) \max(1, \|BF^{-1}\|_\infty).$$

If $\mathcal{A}_T = \mathcal{V}_T \Lambda_T \mathcal{V}_T^{-1}$ *is diagonalizable, then for any eigenvalue* $\mu \in \sigma(\hat{\mathcal{A}}_T)$,

$$\min_{\lambda \in \sigma(\mathcal{A}_T)} |\lambda - \mu| \leq \|E\hat{F}^{-1}\|_\infty \kappa_\infty(\mathcal{V}_T)\, (1 + \|B^t Q^{-1}\|_\infty) \max(1, \|BF^{-1}\|_\infty).$$

Proof. The result is an immediate consequence of the Bauer–Fike theorem [9, p. 342], which states that for diagonalizable $\mathcal{A} = \mathcal{V}\Lambda\mathcal{V}^{-1}$, any $\mu \in \sigma(\mathcal{A} + \mathcal{E})$ satisfies $\min_{\lambda \in \sigma(\mathcal{A})} |\lambda - \mu| \leq \kappa(\mathcal{V})\, \|\mathcal{E}\|$, where $\|\cdot\|$ is any l_p-norm. □

Thus, if $\hat{F}$ is a good enough approximation to F, i.e., if enough inner iterations are used, then $\|E\hat{F}^{-1}\|$ will be small and the eigenvalues of $\hat{\mathcal{A}}_D$ and $\hat{\mathcal{A}}_T$ will be close to those of $\mathcal{A}_D$ and $\mathcal{A}_T$, respectively. We state the result in terms of the l_∞-norm only because the bounds then have a simple form.

Remark 4. We have computed the condition numbers $\kappa(\mathcal{V})$ for $\mathcal{A}_T$ and found them to be large, on the order of 10^3 or higher, for the three values of ν, with streamline upwinding and $h = 1/16$. However, the presence of $\kappa(\mathcal{V})$ in these bounds is an artifact of the proof of the Bauer–Fike theorem; there are more subtle analyses ([9, pp. 344ff]), as well as bounds that do not require diagonalizable matrices [11]. We have observed that the eigenvalues of $\mathcal{A}_T$ are insensitive to perturbations, and we believe that the presence of $\kappa(\mathcal{V})$ is pessimistic. This supposition is supported by the experimental results described below.

To demonstrate that inexact preconditioning is effective, we consider versions of it based on two line-oriented splittings of F. The first uses a *horizontal line Gauss–Seidel relaxation*: Let $F = H - R$ denote a horizontal line Gauss–Seidel splitting of the block convection-diffusion operator F derived from the one-line natural left-to-right, bottom-to-top ordering of the velocity grid. Thus, H is a block lower triangular matrix consisting of the block diagonal of F (a tridiagonal matrix) together with the strict block lower triangular part of F. (See [18], [22] for further details.) The horizontal line Gauss–Seidel method for $Fw = v$ performs the iteration

$$w_0 = 0, \quad w_{i+1} = w_i + H^{-1}(v - Fw_i).$$

For k steps of this iteration, the approximating matrix is $\hat{F} = F(I - (H^{-1}R)^k)^{-1}$.

It has been observed that the performance of relaxation methods of this type can be improved if the sweep direction follows the underlying direction of flow [4]. Our benchmark problem has a circular flow, so that no simple line relaxation can mimic the flow direction throughout Ω. A slightly more sophisticated idea is to use an *alternating line relaxation.* For this, let $F = V - T$ denote a vertical line Gauss–Seidel splitting of F; that is, if P is a permutation matrix associated with the mapping from the natural horizontal line ordering of grid points to the natural vertical line ordering, then P^TVP is the block lower triangular part of P^TFP. One iteration of alternating line relaxation consists of two line Gauss–Seidel steps, one using the horizontal splitting, followed by one using the vertical splitting:

$$w_0 = 0, \quad w_{i+1/2} = w_i + H^{-1}(v - Fw_i), \quad w_{i+1} = w_{i+1/2} + V^{-1}(v - Fw_{i+1/2}).$$

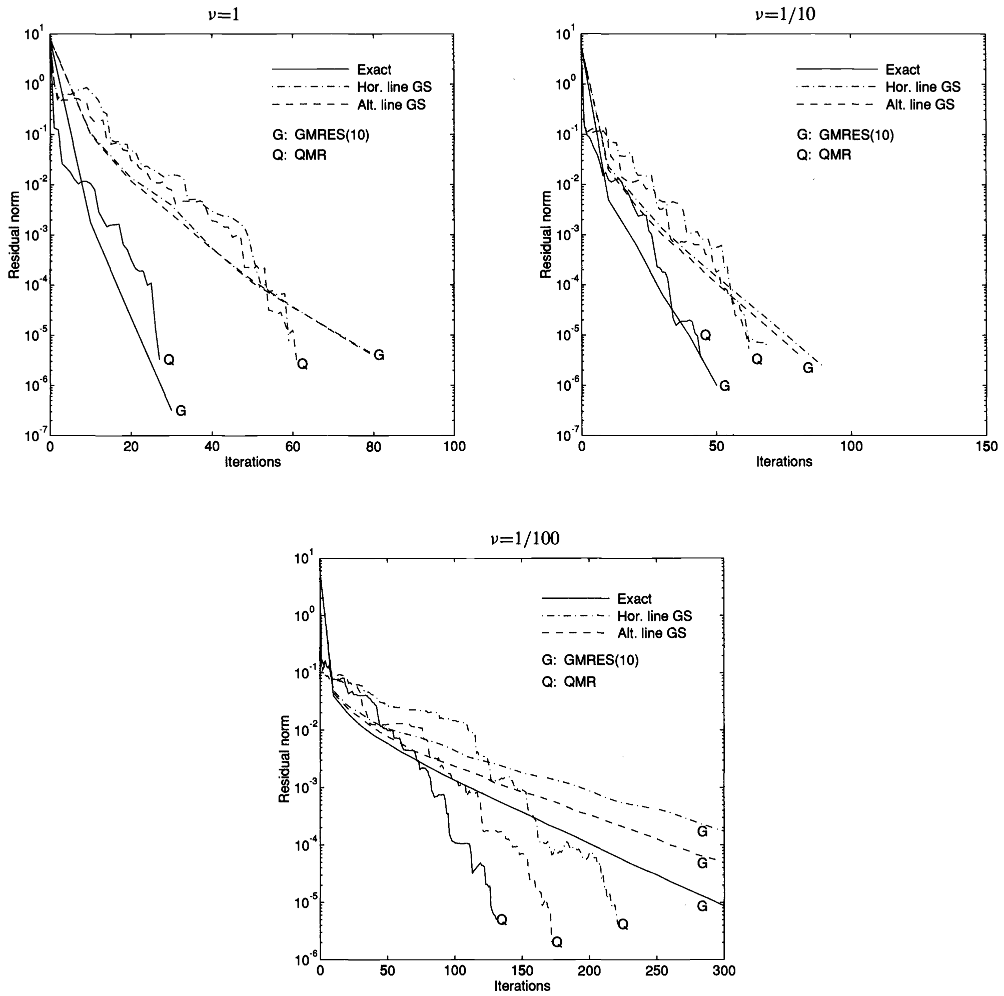


FIG. 6. *Performance of block tridiagonal inexact preconditioners for* 32 × 32 *grid.*

Figure 6 shows results of using the inexact block tridiagonal preconditioners with a version of GMRES with restarts every ten steps (denoted GMRES(10)) and with QMR. The test problem is discretized by streamline upwinding on a 32 × 32 grid. Results for inexact block diagonal preconditioners were similar, except that, as with the exact preconditioners, convergence was slower. We used four steps of horizontal line relaxation or two steps of alternating line relaxation, so that both inexact preconditioners perform four sweeps. The figure also shows the performance of the exact preconditioner, whose cost per step is significantly more expensive. For example, with an $n \times n$ velocity grid, direct solution using a bandsolver requires $O(n^4)$ operations, whereas each inner iteration is an $O(n^2)$ computation. We see that the use of inexact preconditioners in place of the exact versions leads to little degradation of performance of the Krylov subspace methods. For example, in the convection-dominated case $\nu = 1/100$, QMR with alternating line relaxation requires roughly 25% more iterations than with the exact preconditioner. For the diffusion dominated case $\nu = 1$, roughly three times as many outer iterations are required with the inexact preconditioners, which still leads to a less costly computation. Not surprisingly, alternating relaxation is more effective than horizontal relaxation, especially for convection-dominated problems. We remark that our goal here is only to demonstrate "proof-of-concept"; many other techniques for approximating the action of F^{-1} are possible, for both diffusion-dominated and convection-dominated flow. See, for example, [1], [4], [14], [16], [20].

Remark 5. Although we do not make a detailed comparison of Krylov subspace methods, we briefly comment on the behavior of the two choices used here. QMR requires twice as many preconditioned matrix–vector products per step as GMRES(10), and since matrix–vector products are the dominant cost, each QMR step is roughly twice as expensive. Thus, these results indicate that GMRES(10) is more efficient than QMR for large ν, but QMR becomes more effective as convection becomes dominant. The storage requirements of the two methods are comparable.

Acknowledgments. We thank Oliver Ernst, Roland Freund, and Marlis Hochbruck for providing us with Matlab versions of QMR.

REFERENCES

[1] O. Axelsson, V. Eijkhout, B. Polman, and P. Vassilevski, *Incomplete block-matrix factorization iterative methods for convection-diffusion problems*, BIT, 29 (1989), pp. 867–889.

[2] F. Brezzi and M. Fortin, *Mixed and Hybrid Finite Element Methods*, Springer-Verlag, New York, 1991.

[3] A. Brooks and T. Hughes, *Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations*, Comp. Methods Appl. Mech. Engrg., 32 (1982), pp. 199–259.

[4] H. C. Elman and G. H. Golub, *Line iterative methods for cyclically reduced convection-diffusion problems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 339–363.

[5] H. C. Elman and M. H. Schultz, *Preconditioning by fast direct methods for nonselfadjoint nonseparable elliptic problems*, SIAM J. Numer. Anal, 23 (1986), pp. 44–57.

[6] B. Fischer, A. Ramage, D. Silvester, and A. J. Wathen, *Minimum Residual Methods for Augmented Systems*, Tech. Report 15, Department of Mathematics, Strathclyde University, Glasgow, Scotland, 1995.

[7] R. Freund and N. M. Nachtigal, *QMR: A quasi-minimal residual method for non-Hermitian linear systems*, Numer. Math., 60 (1991), pp. 315–339.

[8] ———, *An implementation of the QMR method based on coupled two-term recurrences*, SIAM J. Sci. Comput., 15 (1994), pp. 313–337.

[9] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 2nd ed., The Johns Hopkins University Press, Baltimore, MD, 1989.

[10] M. Gunzburger, *Finite Element Methods for Viscous Incompressible Flows*, Academic Press, San Diego, 1989.

[11] P. Henrici, *Bounds for iterates, inverses, spectral variation, and fields of values of non-normal matrices*, Numer. Math., 4 (1962), pp. 24–40.

[12] C. Johnson, *Numerical Solution of Partial Differential Equations by the Finite Element Method*, Cambridge University Press, New York, 1987.

[13] K. W. Morton, A. Priestley, and E. Süli, *Stability of the Lagrange-Galerkin method with non-exact integration*, Math. Modelling Numer. Anal., 22 (1988), pp. 625–653.

[14] T. Rusten and R. Winther, *A preconditioned iterative method for saddle point problems*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 887–904.

[15] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[16] D. Silvester and A. Wathen, *Fast iterative solution of stabilised Stokes systems part II: Using block preconditioners*, SIAM J. Numer. Anal., 31 (1994), pp. 1352–1367.

[17] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, 2nd ed., Springer-Verlag, New York, 1993.

[18] R. S. Varga, *Matrix Iterative Analysis*, Prentice–Hall, Englewood Cliffs, NJ, 1962.

[19] R. Verfürth, *A combined conjugate gradient-multigrid algorithm for the numerical solution of the Stokes problem*, IMA J. Numer. Anal., 4 (1984), pp. 441–455.

[20] A. Wathen and D. Silvester, *Fast iterative solution of stabilized Stokes systems. Part I: Using simple diagonal preconditioners*, SIAM J. Numer. Anal., 30 (1993), pp. 630–649.

[21] A. J. Wathen, *Realistic eigenvalue bounds for the Galerkin mass matrix*, IMA J. Numer. Anal., 7 (1987), pp. 449–457.

[22] D. M. Young, *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1970.

SIAM J. SCI. COMPUT.
Vol. 17, No. 1, pp. 47–64, January 1996

004

ANALYSIS OF SEMI-TOEPLITZ PRECONDITIONERS FOR FIRST-ORDER PDES*

LINA HEMMINGSSON† AND KURT OTTO†

Abstract. A semi-Toeplitz preconditioner for nonsymmetric, nondiagonally dominant systems of equations is studied. The preconditioner solve is based on a fast modified sine transform. As a model problem we study a system of equations arising from an implicit time discretization of a scalar hyperbolic partial differential equation (PDE). Analytical formulas for the eigenvalues and the eigenvectors of the preconditioned system are derived. The convergence of a minimal residual iteration is shown to depend only on the spatial grid ratio and not on the number of unknowns.

Key words. first-order PDE, finite difference discretization, preconditioner, spectral analysis, convergence

AMS subject classifications. 65F10, 65N22

1. Introduction.

We are interested in solving systems of first-order partial differential equations (PDEs) such as the Euler equations and the Maxwell equations. For the Euler equations, we are mainly interested in the case when there are different time scales present in the problem, and the fastest time scale does not have to be resolved. An important application is almost incompressible flow [4], [5], where the sound waves in the medium are much faster than the motion of the fluid. For an explicit discretization in time, the restriction on the time step due to the Courant–Friedrichs–Lewy criterion becomes too severe. For the Maxwell equations, we intend to consider complicated geometries such that the smallest grid cell in the space discretization restricts the time step considerably in an explicit time discretization. Therefore, we use an implicit time discretization. This leads to a system of equations

$$Au = b, \tag{1}$$

which has to be solved at each time step. A is nonsymmetric, and since the time step is large compared to the space step, it is also strongly nondiagonally dominant.

In this paper we examine theoretically the convergence properties when we apply a minimal residual iteration to (1), utilizing a semi-Toeplitz preconditioner. Such preconditioners have been shown to be well suited in a domain decomposition setting of the problem [8].

2. The model problem.

2.1. The differential equation.

In [7] we consider a scalar two-dimensional equation with variable coefficients. Here we will restrict our presentation to constant coefficients in order to be able to derive theoretical properties of the preconditioned system.

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x_1} + \frac{\partial u}{\partial x_2} = g, \quad 0 < x_1 \leq 1, \quad 0 < x_2 \leq 1, \quad t > 0. \tag{2}$$

Problem (2) is well posed if we prescribe $u(0, x_2, t)$, $u(x_1, 0, t)$, and $u(x_1, x_2, 0)$.

2.2. Discretization.

Introduce a uniform grid as

$$x_{d,j} = jh_d, \quad j = 0, \ldots, m_d, \quad h_d = 1/m_d, \quad d = 1, 2.$$

*Received by the editors May 9, 1994; accepted for publication February 25, 1995. This work was supported by the Swedish National Board for Industrial and Technical Development (NUTEK) under contract No. 9303806.

†Department of Scientific Computing, Uppsala University, Box 120, S-751 04 Uppsala, Sweden (lina@tdb.uu.se and kurt@tdb.uu.se).

Let $u_{j,k}$ denote the approximate solution at the point $(x_{1,j}, x_{2,k})$. Now we discretize equation (2) in time using the trapezoidal rule with time step Δt:

$$\frac{u^{n+1} - u^n}{\Delta t} + \tfrac{1}{2}\sum_{d=1}^{2}\left(\left(\frac{\partial u}{\partial x_d}\right)^{n+1} + \left(\frac{\partial u}{\partial x_d}\right)^{n}\right) = \tfrac{1}{2}(g^{n+1} + g^n).$$

For the spatial derivatives, we use second-order centered differences in the interior of the domain and one-sided first-order differences at the outflow boundaries. We define the following quantities:

$$\kappa_d \equiv \frac{\Delta t}{h_d}, \quad d = 1, 2,$$

$$u^n = \begin{pmatrix} u^n_{1,1} & u^n_{2,1} & \cdots & u^n_{m_1,1} & u^n_{1,2} & \cdots & u^n_{m_1,m_2} \end{pmatrix}^T,$$

and finally write the equations as

$$Au^{n+1} \equiv (I_{m_2} \otimes A_1 + \kappa_2 A_2 \otimes I_{m_1})u^{n+1} = b, \tag{3}$$

where b contains known quantities, and A_1 and A_2 are defined by

$$A_1 = \begin{pmatrix} 4 & \kappa_1 & & & \\ -\kappa_1 & 4 & \kappa_1 & & \\ & \ddots & \ddots & \ddots & \\ & & -\kappa_1 & 4 & \kappa_1 \\ & & & -2\kappa_1 & 4 + 2\kappa_1 \end{pmatrix}, \tag{4}$$

$$A_2 = \begin{pmatrix} 0 & 1 & & & \\ -1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 0 & 1 \\ & & & -2 & 2 \end{pmatrix}.$$

3. Iterative methods. The matrix A defined in (3) is extremely sparse, with five nonzero diagonals and semibandwidth m_1. Modern CG-like iterative methods are well suited for nonsymmetric systems of equations. In this paper we consider minimal residual iterations [3], which in each iteration satisfy

$$\|r^{(i)}\|_2 = \min_{p_i \in \mathcal{P}_i,\, p_i(0)=1} \|p_i(M^{-1}A)r^{(0)}\|_2,$$

where $\mathcal{P}_i$ is the set of all polynomials of degree i, M is the left preconditioner, $r^{(i)} \equiv M^{-1}(Au^{(i)} - b)$, and $u^{(i)}$ denotes the approximation of u obtained at iteration i. If $M^{-1}A$ is diagonalizable we obtain

$$\frac{\|r^{(i)}\|_2}{\|r^{(0)}\|_2} \le \mathrm{cond}_2(W_{M^{-1}A}) \min_{p_i \in \mathcal{P}_i,\, p_i(0)=1} \max_{1 \le \ell \le n} |p_i(\lambda_\ell)| \equiv \mathrm{cond}_2(W_{M^{-1}A}) \cdot \varepsilon_i, \tag{5}$$

where $W_{M^{-1}A}$ is the eigenvector matrix and λ_ℓ are the eigenvalues of $M^{-1}A$. From (5) we conclude that we will have finite termination in n iterations, where n is the number of distinct eigenvalues to the preconditioned system. Moreover, if we can precondition our system in such a way that the eigenvalues of $M^{-1}A$ are contained in k dense clusters, we have a good approximation to the solution in k iterations. Clustering of the eigenvalues may be even more important than a condition number improvement [1], [2], and [12]. In §4 we define a preconditioner that yields a highly clustered spectrum. Since the iterative method includes the solution of

$$Mx = y \tag{6}$$

in each step, we must have a fast solver for this system. The preconditioner solve defined in this paper is based on a fast modified sine transform developed in [6].

4. Preconditioning. Consider the semi-Toeplitz preconditioner M defined by

$$M = I_{m_2} \otimes \hat{A}_1 + \kappa_2 A_2 \otimes I_{m_1}, \quad \hat{A}_1 = \begin{pmatrix} 4 & \kappa_1 & & \\ -\kappa_1 & \ddots & \ddots & \\ & \ddots & \ddots & \kappa_1 \\ & & -\kappa_1 & 4 \end{pmatrix}. \tag{7}$$

In [7] it is shown that M can be decomposed as

$$M = (I_{m_2} \otimes S_{m_1}) T (I_{m_2} \otimes S_{m_1}^H),$$

where T is block tridiagonal with diagonal blocks

$$T = I_{m_2} \otimes \Lambda_1 + \kappa_2 A_2 \otimes I_{m_1},$$

where

$$\Lambda_1 = \operatorname{diag}(\lambda_{1,1}, \ldots, \lambda_{1,m_1}), \quad \lambda_{1,j} = 4 + 2i\kappa_1 \cos\left(\frac{j\pi}{m_1+1}\right). \tag{8}$$

S_{m_1} is the modified sine matrix defined by

$$S_{m_1}(j,k) = \sqrt{\frac{2}{m_1+1}}\, i^{j+k+1} \sin\left(\frac{jk\pi}{m_1+1}\right), \quad j,k = 1,\ldots,m_1. \tag{9}$$

By rearranging the unknowns, the solution of

$$Tx = y$$

decouples into m_1 independent tridiagonal systems of order m_2. In [6] it is shown how the modified sine transforms can be evaluated using Fourier transforms of vectors $y \in C^{\frac{m_1+1}{2}}$. By symmetry we can also reduce the intermediate solution of the tridiagonal systems to the solution of only the first $\frac{m_1+1}{2}$ systems.

To sum up this section we present the different steps in the solution of (6):

- m_2 Fourier transforms of vectors $y \in C^{\frac{m_1+1}{2}}$,
- $\frac{m_1+1}{2}$ solutions of tridiagonal systems of order m_2,
- m_2 Fourier transforms of vectors $y \in C^{\frac{m_1+1}{2}}$.

5. Spectral analysis. By defining

$$E \equiv A - M = I_{m_2} \otimes E_1,$$

where E_1 is defined by

$$E_1 = \begin{pmatrix} 0 & \cdots & 0 & 0 & 0 \\ \vdots & & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & 0 \\ 0 & \cdots & 0 & -\kappa_1 & 2\kappa_1 \end{pmatrix},$$

we get

$$M^{-1}A = I + M^{-1}(I_{m_2} \otimes E_1). \tag{10}$$

Otto [9] shows how to compute the eigenvalues and the eigenvectors of (10) when M is a semicirculant preconditioner, i.e., each block is circulant. We use a similar technique here. We will make use of a lemma in [9], which we restate for readability. Let $\mathcal{E}(a, b)$ denote the closed ellipse centered at the origin with semimajor axis b oriented along the imaginary axis and semiminor axis a. Also, let $\mathcal{E}^+(a, b)$ denote the region $\{z | z \in \mathcal{E}(a, b) \text{ and } \Re e(z) \geq 0\}$.

LEMMA 5.1. *The eigenvalues* $\lambda_{2,k}$, $k = 1, \ldots, m_2$, *of* A_2 *satisfy*

- $\lambda_{2,k} \neq \lambda_{2,j}, \quad k \neq j,$
- $\lambda_{2,k} \in \mathcal{E}^+(4m_2^{-3/4}, 2 + 4m_2^{-3/2})$.

From Lemma 5.1 we conclude that there exists a nonsingular matrix V_2 such that

$$V_2^{-1} A_2 V_2 = \Lambda_2 = \mathrm{diag}(\lambda_{2,1}, \ldots, \lambda_{2,m_2}).$$

By defining

$$D \equiv (V_2^{-1} \otimes S_{m_1}^H) M (V_2 \otimes S_{m_1}),$$

where S_{m_1} is the modified sine matrix of order m_1 defined in (9), we get

$$\begin{aligned} D &= (V_2^{-1} \otimes S_{m_1}^H)(I_{m_2} \otimes \hat{A}_1 + \kappa_2 A_2 \otimes I_{m_1})(V_2 \otimes S_{m_1}) \\ &= I_{m_2} \otimes S_{m_1}^H \hat{A}_1 S_{m_1} + \kappa_2 V_2^{-1} A_2 V_2 \otimes I_{m_1} = I_{m_2} \otimes \Lambda_1 + \kappa_2 \Lambda_2 \otimes I_{m_1}, \end{aligned}$$

where Λ_1 is defined in (8). From Lemma 5.1 we get that

$$\Re e(\lambda_{1,j} + \kappa_2 \lambda_{2,k}) = 4 + \kappa_2 \Re e(\lambda_{2,k}) \geq 4,$$

which implies that $I_{m_2} \otimes \Lambda_1 + \kappa_2 \Lambda_2 \otimes I_{m_1}$ is nonsingular, i.e., M^{-1} exists. Hence,

$$D = I_{m_2} \otimes \Lambda_1 + \kappa_2 \Lambda_2 \otimes I_{m_1} = \mathrm{diag}(D_1, \ldots, D_{m_2}),$$

where

$$D_k = \mathrm{diag}(d_\nu),$$

$$d_\nu = 4 + 2i\kappa_1 \cos\left(\frac{j\pi}{m_1+1}\right) + \kappa_2 \lambda_{2,k}, \qquad \begin{cases} \nu = (k-1)m_1 + j, \\ k = 1, \ldots, m_2, \\ j = 1, \ldots, m_1, \end{cases} \tag{11}$$

and the spectral decomposition of M^{-1} becomes

$$M^{-1} = (V_2 \otimes S_{m_1}) D^{-1} (V_2^{-1} \otimes S_{m_1}^H).$$

Now consider the matrix W given by

$$\begin{aligned} W &= (V_2 \otimes I_{m_1})^{-1} M^{-1} E (V_2 \otimes I_{m_1}) \\ &= (I_{m_2} \otimes S_{m_1}) \mathrm{diag}(D_1^{-1}, \ldots, D_{m_2}^{-1})(I_{m_2} \otimes S_{m_1}^H E_1) \\ &= \mathrm{diag}(S_{m_1} D_1^{-1} S_{m_1}^H E_1, \ldots, S_{m_1} D_{m_2}^{-1} S_{m_1}^H E_1) \equiv \mathrm{diag}(W_1, \ldots, W_{m_2}). \end{aligned}$$

As W is a similarity transformation of $M^{-1}E$ and W is block diagonal, the eigenvalues of $M^{-1}E$ are equal to the eigenvalues of W_k, $k = 1, \ldots, m_2$. Due to the sparse structure of E_1, W_k only has two nonzero columns,

$$W_k = \begin{pmatrix} 0 & \cdots & 0 & W_k(1, m_1 - 1) & W_k(1, m_1) \\ 0 & \cdots & 0 & W_k(2, m_1 - 1) & W_k(2, m_1) \\ \vdots & & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & W_k(m_1, m_1 - 1) & W_k(m_1, m_1) \end{pmatrix}, \tag{12}$$

where

$$\begin{aligned} W_k(\ell, m_1 - 1) &= -\kappa_1 \sum_{j=1}^{m_1} S_{m_1}(\ell, j) d_\nu^{-1} S_{m_1}^H(j, m_1) \\ &= -\kappa_1 i^{\ell - m_1} \Phi_{m_1,\ell}(\tfrac{4+\kappa_2\lambda_{2,k}}{2}, \kappa_1), \\ W_k(\ell, m_1) &= 2\kappa_1 \sum_{j=1}^{m_1} S_{m_1}(\ell, j) d_\nu^{-1} S_{m_1}^H(j, m_1) \\ &= 2\kappa_1 i^{\ell - m_1} \Phi_{m_1,\ell}(\tfrac{4+\kappa_2\lambda_{2,k}}{2}, \kappa_1), \qquad \ell = 1, \ldots, m_1, \end{aligned} \tag{13}$$

where $\nu = (k-1)m_1 + j$, d_ν is defined in (11) and

$$\Phi_{k,\ell}(\alpha, \beta) = \frac{2}{m_1 + 1} \sum_{j=1}^{m_1} F_{k,\ell}\left(\frac{j\pi}{m_1+1}\right), \tag{14}$$

$$F_{k,\ell}(\theta) = \frac{\sin(k\theta)\sin(\ell\theta)}{2\alpha + 2i\beta\cos(\theta)}, \qquad k, \ell = 1, \ldots, m_1, \tag{15}$$

where α is a complex number with $\Re e(\alpha) > 0$, and $\beta > 0$ is real. We now show how to evaluate the sums in (14). Using this result we can then compute the sums defined in (13).

LEMMA 5.2. *$\Phi_{k,\ell}(\alpha, \beta)$ defined in* (14) *can be evaluated by*

$$\Phi_{k,\ell}(\alpha, \beta) = \frac{i}{\beta}\left(\frac{z^{k+\ell} - z^{|k-\ell|}}{z - z^{-1}} + \frac{(z^k - z^{-k})(z^\ell - z^{-\ell})}{z - z^{-1}} \frac{z^{2m_1+2}}{1 - z^{2m_1+2}}\right),$$

where

$$z = i\left(\tfrac{\alpha}{\beta} - \sqrt{1 + \left(\tfrac{\alpha}{\beta}\right)^2}\right)$$

and the principal branch of the square root function is employed.

Proof. By periodicity we get

$$\Phi_{k,\ell}(\alpha,\beta) = \frac{1}{m_1+1}\sum_{j=0}^{2m_1+1} F_{k,\ell}\left(\tfrac{j\pi}{m_1+1}\right), \quad k,\ell = 1,\ldots,m_1.$$

$F_{k,\ell}$ is C^1 and 2π-periodic, implying

$$F_{k,\ell}(\theta) = \sum_{q=-\infty}^{\infty} c_q e^{iq\theta}, \quad c_q = \frac{1}{2\pi}\int_0^{2\pi} e^{-iq\theta}F_{k,\ell}(\theta)d\theta.$$

The Poisson summation formula yields

$$\Phi_{k,\ell}(\alpha,\beta) = \sum_{q=-\infty}^{\infty} 2c_{q(2m_1+2)} = 2c_0 + \sum_{q=1}^{\infty} 2c_{-q(2m_1+2)} + 2c_{q(2m_1+2)}.$$

Using the Euler identities in (15) gives

$$\begin{aligned} F_{k,\ell}(\theta) &= -\frac{1}{4}\frac{(e^{ik\theta}-e^{-ik\theta})(e^{i\ell\theta}-e^{-i\ell\theta})}{2\alpha+i\beta(e^{i\theta}+e^{-i\theta})} \\ &= \frac{e^{i(k-\ell)\theta}+e^{i(\ell-k)\theta}-e^{i(k+\ell)\theta}-e^{-i(k+\ell)\theta}}{8\alpha+4i\beta(e^{i\theta}+e^{-i\theta})}. \end{aligned}$$

By making the substitution $z = e^{i\theta}$ we get

$$c_q = -\frac{i}{4\beta}\frac{1}{2\pi i}\oint_C \frac{z^{k-\ell-q}+z^{\ell-k-q}-z^{k+\ell-q}-z^{-k-\ell-q}}{(z-z_1)(z-z_2)}dz, \tag{16}$$

where C is the positively oriented unit circle and

$$z_1 = i\left(\tfrac{\alpha}{\beta}-\sqrt{1+\left(\tfrac{\alpha}{\beta}\right)^2}\right), \quad z_2 = i\left(\tfrac{\alpha}{\beta}+\sqrt{1+\left(\tfrac{\alpha}{\beta}\right)^2}\right), \tag{17}$$

where $|z_1| < 1$ and $|z_2| > 1$. Hence, z_1 is within C while z_2 is not. Note that $z_2 = z_1^{-1}$.

Two types of integrals arise, which can be evaluated using the residue theorem.

- $p \geq 0$, $I_1 = \frac{1}{2\pi i}\oint_C \frac{z^p}{(z-z_1)(z-z_2)}dz = \frac{z_1^p}{z_1-z_2} = \frac{z_1^p}{z_1-z_1^{-1}}$,
- $p \geq 1$, $I_2 = \frac{1}{2\pi i}\oint_C \frac{z^{-p}}{(z-z_1)(z-z_2)}dz = \frac{z_1^{-p}}{z_1-z_2} + \frac{1}{(p-1)!}\varphi^{(p-1)}(0)$.

Here $\varphi^{(p-1)}(0)$ denotes the $(p-1)$th derivative of $\varphi(z)$ at $z = 0$, where $\varphi(z)$ is defined by

$\varphi(z) = \frac{1}{(z-z_1)(z-z_2)} = \frac{1}{z_1-z_2}\left((z-z_1)^{-1}-(z-z_2)^{-1}\right)$, yielding

$\varphi^{(p-1)}(z) = \frac{1}{z_1-z_2}(p-1)!(-1)^{p-1}\left((z-z_1)^{-p}-(z-z_2)^{-p}\right)$, and hence

$$I_2 = \frac{z_1^{-p}}{z_1-z_1^{-1}} + \frac{(-1)^{p-1}}{z_1-z_1^{-1}}\left((-z_1)^{-p}-(-z_2)^{-p}\right) = \frac{z_1^p}{z_1-z_1^{-1}}.$$

Using this result in (16) yields

$$\begin{cases} c_0 = \frac{i}{2\beta}\left(\frac{z_1^{k+\ell}-z_1^{|k-\ell|}}{z_1-z_1^{-1}}\right), & \\ c_{-q(2m_1+2)} = \frac{i}{4\beta}\left(\frac{z_1^{k+\ell}+z_1^{-k-\ell}-z_1^{k-\ell}-z_1^{\ell-k}}{z_1-z_1^{-1}}\right)z_1^{q(2m_1+2)}, & k,\ell = 1,\dots,m_1, \\ c_{q(2m_1+2)} = c_{-q(2m_1+2)}. & \end{cases}$$

Since $|z_1| < 1$, we get

$$\sum_{q=1}^{\infty}(z_1^{2m_1+2})^q = \frac{z_1^{2m_1+2}}{1-z_1^{2m_1+2}},$$

which gives

$$\sum_{q=1}^{\infty} 2c_{-q(2m_1+2)} + 2c_{q(2m_1+2)} = \frac{i}{\beta}\frac{z_1^{k+\ell}+z_1^{-k-\ell}-z_1^{k-\ell}-z_1^{\ell-k}}{z_1-z_1^{-1}}\frac{z_1^{2m_1+2}}{1-z_1^{2m_1+2}}.$$

Summing up, we conclude that

$$\Phi_{k,\ell}(\alpha,\beta) = \frac{i}{\beta}\left(\frac{z_1^{k+\ell}-z_1^{|k-\ell|}}{z_1-z_1^{-1}} + \frac{(z_1^k-z_1^{-k})(z_1^\ell-z_1^{-\ell})}{z_1-z_1^{-1}}\frac{z_1^{2m_1+2}}{1-z_1^{2m_1+2}}\right),$$

where z_1 is defined in (17), which completes the proof. □

Thus, by Lemma 5.2 and the facts that $\Re e(4+\kappa_2\lambda_{2,k}) > 0$ and $\kappa_1 > 0$, we can find analytical expressions for $W_k(\ell, m_1-1)$ and $W_k(\ell, m_1)$ defined in (13). The characteristic equation for W_k defined in (12) is

$$\begin{aligned} 0 &= \det(\lambda I_{m_1} - W_k) \\ &= \lambda^{m_1-2}\begin{vmatrix} \lambda - W_k(m_1-1,m_1-1) & -W_k(m_1-1,m_1) \\ -W_k(m_1,m_1-1) & \lambda - W_k(m_1,m_1) \end{vmatrix} \\ &= \lambda^{m_1-1}\left(\lambda - i\kappa_1\Phi_{m_1,m_1-1}(\tfrac{4+\kappa_2\lambda_{2,k}}{2},\kappa_1) - 2\kappa_1\Phi_{m_1,m_1}(\tfrac{4+\kappa_2\lambda_{2,k}}{2},\kappa_1)\right). \end{aligned}$$

Hence, W_k only has one nonzero eigenvalue λ_k, given by

$$\begin{aligned} \lambda_k &= i\kappa_1\Phi_{m_1,m_1-1}(\tfrac{4+\kappa_2\lambda_{2,k}}{2},\kappa_1) + 2\kappa_1\Phi_{m_1,m_1}(\tfrac{4+\kappa_2\lambda_{2,k}}{2},\kappa_1) \\ &= -z_k^2\frac{1-z_k^{2m_1-2}}{1-z_k^{2m_1+2}} + 2iz_k\frac{1-z_k^{2m_1}}{1-z_k^{2m_1+2}}, \end{aligned}$$

where

$$z_k = i\left(\tfrac{4+\kappa_2\lambda_{2,k}}{2\kappa_1} - \sqrt{1+\left(\tfrac{4+\kappa_2\lambda_{2,k}}{2\kappa_1}\right)^2}\right). \tag{18}$$

We summarize this result in a theorem.

THEOREM 5.3. *The matrix $M^{-1}A$, where M and A are defined in* (7) *and* (3), *has $(m_1-1)m_2$ eigenvalues that are identically one. The remaining m_2 eigenvalues μ_k are given by*

$$\mu_k = 1 - z_k^2\frac{1-z_k^{2m_1-2}}{1-z_k^{2m_1+2}} + 2iz_k\frac{1-z_k^{2m_1}}{1-z_k^{2m_1+2}}, \quad k = 1,\dots,m_2, \tag{19}$$

where z_k is defined in (18).

As the dimension of the nullspace of E is $(m_1 - 1)m_2$, it is clear that the eigenvalue zero of $M^{-1}E$ has $(m_1 - 1)m_2$ linearly independent eigenvectors. Hence, we know that a minimal residual iteration will converge in at most $m_2 + 1$ iterations. We now derive the eigenvectors of $M^{-1}A$.

THEOREM 5.4. *Assume that the nontrivial eigenvalues λ_k, $k = 1, \ldots, m_2$, defined in* (18), *are nonzero. Then the matrix $M^{-1}A$ has a nonsingular eigenvector matrix*

$$W_{M^{-1}A} = (V_2 \otimes I_{m_1})\operatorname{diag}(U_1, \ldots, U_{m_2}).$$

Here V_2 is the eigenvector matrix of A_2 and U_k is given by

$$U_k = \left[e_1, \ldots, e_{m_1-2}, (2e_{m_1-1} + e_{m_1})/\sqrt{5}, u^{(k)}/\|u^{(k)}\|_2\right], \quad k = 1, \ldots, m_2,$$

where e_j denotes the jth canonical column vector and

$$u_\ell^{(k)} = i^\ell(z_k^\ell - z_k^{-\ell}), \quad \ell = 1, \ldots, m_1,$$

where z_k is defined in (18).

Proof. In the proof of Theorem 5.3 we derived the formula

$$(V_2 \otimes I_{m_1})^{-1} M^{-1} E (V_2 \otimes I_{m_1}) = \operatorname{diag}(W_1, \ldots, W_{m_2}),$$

where the eigenvalues of W_k are λ_k and zero with multiplicity $m_1 - 1$. The first $m_1 - 2$ columns of W_k are identically zero, and combining (13) and Lemma 5.2 yields

$$\left.\begin{aligned} W_k(\ell, m_1 - 1) &= \frac{i^{-m_1+1} z_k^{m_1+1}}{1 - z_k^{2m_1+2}} i^\ell (z_k^\ell - z_k^{-\ell}) \\ W_k(\ell, m_1) &= -2W_k(\ell, m_1 - 1). \end{aligned}\right\}, \quad \ell = 1, \ldots, m_1,$$

We now investigate the eigenvector matrix U_k of W_k. Due to the structure of W_k we conclude that $\left\{\{e_j\}_{j=1}^{m_1-2}, (2e_{m_1-1} + e_{m_1})/\sqrt{5}\right\} \in \operatorname{null}(W_k)$. Consequently, those $m_1 - 1$ orthonormal vectors are eigenvectors of W_k corresponding to the eigenvalue zero. The assumption $\lambda_k \neq 0$ implies that the corresponding eigenvector $u^{(k)} \notin \operatorname{null}(W_k)$. Thus, W_k has m_1 linearly independent eigenvectors. From $0 < |z_k| < 1$ it follows that column $m_1 - 1$ of W_k is nonzero. Furthermore, the structure of W_k implies that column $m_1 - 1$ is an eigenvector corresponding to λ_k. By rescaling the column we can choose

$$u_\ell^{(k)} = i^\ell(z_k^\ell - z_k^{-\ell}), \quad \ell = 1, \ldots, m_1,$$

as the nontrivial eigenvector to W_k. We then obtain

$$U_k^{-1} W_k U_k = \Lambda_k = \operatorname{diag}(0, \ldots, 0, \lambda_k), \quad k = 1, \ldots, m_2.$$

Now the matrix $W_{M^{-1}A} \equiv (V_2 \otimes I_{m_1})\operatorname{diag}(U_1, \ldots, U_{m_2})$ is nonsingular and

$$\begin{aligned} &W_{M^{-1}A}^{-1} M^{-1} A W_{M^{-1}A} \\ &= I + \left(\operatorname{diag}(U_1, \ldots, U_{m_2})\right)^{-1} (V_2 \otimes I_{m_1})^{-1} M^{-1} E (V_2 \otimes I_{m_1}) \operatorname{diag}(U_1, \ldots, U_{m_2}) \\ &= I + \operatorname{diag}(U_1^{-1}, \ldots, U_{m_2}^{-1}) \operatorname{diag}(W_1, \ldots, W_{m_2}) \operatorname{diag}(U_1, \ldots, U_{m_2}) \\ &= I + \operatorname{diag}(\Lambda_1, \ldots, \Lambda_{m_2}), \end{aligned}$$

i.e., $W_{M^{-1}A}$ is the eigenvector matrix of $M^{-1}A$. □

From Theorem 5.6 it is easy to establish that the assumption in Theorem 5.4 is valid, when m_1 and m_2 are sufficiently large. To compare the condition of the eigenvectors for A and $M^{-1}A$, we define the *condition number reduction*

$$\psi \equiv \mathrm{cond}_2(W_{M^{-1}A})\big/\mathrm{cond}_2(W_A), \tag{20}$$

where W_A is the eigenvector matrix of A. Following the technique in [9] ψ can be estimated. Under the same assumption as in Theorem 5.4 we obtain

$$\psi \le \psi_0 = \max_{1\le k\le m_2} \|U_k\|_2 \cdot \max_{1\le k\le m_2} \|U_k^{-1}\|_2 \Big/ \mathrm{cond}_2(V_1), \tag{21}$$

where V_1 is the eigenvector matrix of A_1 defined in (4). Therefore, we determine $\|U_k\|_2$ and $\|U_k^{-1}\|_2$.

THEOREM 5.5. *Under the same assumption as in Theorem* 5.4,

$$\left.\begin{aligned} \|U_k\|_2 &= \left(1+\sqrt{1-a_k}\right)^{1/2} \\ \|U_k^{-1}\|_2 &= \left(1-\sqrt{1-a_k}\right)^{-1/2} \end{aligned}\right., \qquad k = 1, \ldots, m_2,$$

where

$$a_k = \frac{\frac{1}{5}|z_k^{m_1-1} - z_k^{-m_1+1}|^2 + \frac{4}{5}|z_k^{m_1} - z_k^{-m_1}|^2 - \frac{4}{5}\Im m\left((z_k^{m_1-1} - z_k^{-m_1+1})(\bar{z}_k^{m_1} - \bar{z}_k^{-m_1})\right)}{\dfrac{|z_k|^2 - |z_k|^{2m_1+2}}{1-|z_k|^2} + \dfrac{|z_k|^{-2} - |z_k|^{-2m_1-2}}{1-|z_k|^{-2}} - 2\Re e\left(\dfrac{z_k\bar{z}_k^{-1} - (z_k\bar{z}_k^{-1})^{m_1+1}}{1 - z_k\bar{z}_k^{-1}}\right)}.$$

Proof. To derive the results we examine the eigenvalues of $U_k^H U_k$. Using the notation in Theorem 5.4, we define the auxiliary vectors $w^{(k)}$, $k = 1, \ldots, m_2$

$$w_\ell^{(k)} = u_\ell^{(k)}\big/\|u^{(k)}\|_2, \quad \ell = 1, \ldots, m_1 - 2,$$

$$w_{m_1-1}^{(k)} = \left(\tfrac{2}{\sqrt{5}}u_{m_1-1}^{(k)} + \tfrac{1}{\sqrt{5}}u_{m_1}^{(k)}\right)\big/\|u^{(k)}\|_2.$$

Exploiting the orthonormality of the first $m_1 - 1$ columns of U_k yields

$$U_k^H U_k = \begin{pmatrix} I_{m_1-1} & w^{(k)} \\ (w^{(k)})^H & 1 \end{pmatrix}.$$

The characteristic polynomial $P_k(\lambda)$ of $U_k^H U_k$ satisfies

$$\begin{aligned} (1-\lambda)P_k(\lambda) &= (1-\lambda)\det(U_k^H U_k - \lambda I_{m_1}) \\ &= \begin{vmatrix} I_{m_1-1} & 0 \\ -(w^{(k)})^H & 1-\lambda \end{vmatrix} \begin{vmatrix} (1-\lambda)I_{m_1-1} & w^{(k)} \\ (w^{(k)})^H & 1-\lambda \end{vmatrix} \\ &= \begin{vmatrix} (1-\lambda)I_{m_1-1} & w^{(k)} \\ 0 & (1-\lambda)^2 - (w^{(k)})^H w^{(k)} \end{vmatrix} \\ &= (1-\lambda)^{m_1-1}\left((1-\lambda)^2 - \|w^{(k)}\|_2^2\right). \end{aligned}$$

Thus,

$$P_k(\lambda) = (1-\lambda)^{m_1-2}\left((1-\lambda)^2 - \|w^{(k)}\|_2^2\right).$$

Consequently, the eigenvalues of $U_k^H U_k$ are one with multiplicity $m_1 - 2$ and

$$\begin{aligned}\lambda_{k,1} &= 1 + \|w^{(k)}\|_2,\\ \lambda_{k,2} &= 1 - \|w^{(k)}\|_2.\end{aligned}$$

We obtain

$$\left.\begin{aligned}\|U_k\|_2 &= \big(\max(1, \lambda_{k,1}, \lambda_{k,2})\big)^{1/2} = (1 + \|w^{(k)}\|_2)^{1/2}\\ \|U_k^{-1}\|_2 &= \big(\max(1, \lambda_{k,1}^{-1}, \lambda_{k,2}^{-1})\big)^{1/2} = (1 - \|w^{(k)}\|_2)^{-1/2}\end{aligned}\right., \quad k = 1, \dots, m_2.$$

It now remains to derive a formula for $\|w^{(k)}\|_2$. From the definition of $w^{(k)}$ and Theorem 5.4 it follows that

$$\begin{aligned}\|w^{(k)}\|_2^2 &= \sum_{\ell=1}^{m_1-1} |w_\ell^{(k)}|^2 = \|u^{(k)}\|_2^{-2}\big(\tfrac{1}{5}|2u_{m_1-1}^{(k)} + u_{m_1}^{(k)}|^2 + \sum_{\ell=1}^{m_1-2} |u_\ell^{(k)}|^2\big)\\ &= 1 - \|u^{(k)}\|_2^{-2}\big(\tfrac{1}{5}|u_{m_1-1}^{(k)}|^2 + \tfrac{4}{5}|u_{m_1}^{(k)}|^2 - \tfrac{4}{5}\Re e(u_{m_1-1}^{(k)}\bar{u}_{m_1}^{(k)})\big)\\ &= 1 - \|u^{(k)}\|_2^{-2}\Big(\tfrac{1}{5}|z_k^{m_1-1} - z_k^{-m_1+1}|^2 + \tfrac{4}{5}|z_k^{m_1} - z_k^{-m_1}|^2\\ &\qquad -\tfrac{4}{5}\Im m\big((z_k^{m_1-1} - z_k^{-m_1+1})(\bar{z}_k^{m_1} - \bar{z}_k^{-m_1})\big)\Big).\end{aligned}$$

By contradiction one can derive that $z_k \neq \bar{z}_k$. This and $|z_k| \neq 1$ makes $\|u^{(k)}\|_2^2$ computable from the geometric series

$$\begin{aligned}\|u^{(k)}\|_2^2 &= \sum_{\ell=1}^{m_1} |u_\ell^{(k)}|^2 = \sum_{\ell=1}^{m_1} (\bar{z}_k^\ell - \bar{z}_k^{-\ell})(z_k^\ell - z_k^{-\ell})\\ &= \sum_{\ell=1}^{m_1} |z_k^2|^\ell + |z_k^{-2}|^\ell - (z_k\bar{z}_k^{-1})^\ell - (\bar{z}_k z_k^{-1})^\ell\\ &= \frac{|z_k|^2 - |z_k|^{2m_1+2}}{1 - |z_k|^2} + \frac{|z_k|^{-2} - |z_k|^{-2m_1-2}}{1 - |z_k|^{-2}} - 2\Re e\Big(\frac{z_k\bar{z}_k^{-1} - (z_k\bar{z}_k^{-1})^{m_1+1}}{1 - z_k\bar{z}_k^{-1}}\Big),\end{aligned}$$

which completes the proof. □

From Theorem 5.3 we derive the following asymptotic result.

THEOREM 5.6. *Assume that*

$$\Delta t = ch_1^\alpha, \quad 0 < \alpha < 1, \quad c > 0, \tag{22}$$

and

$$m_1 = \left\lceil \frac{m_2(1 + 2m_2^{-\frac{3}{2}})}{\phi} \right\rceil, \quad 0 < \phi < 1, \tag{23}$$

where $\lceil \frac{a}{b} \rceil$ *denotes the closest integer greater than or equal to* $\frac{a}{b}$. *In the limit* $m_1 \to \infty$, *the* m_2 *eigenvalues of* $M^{-1}A$ *that are different from one all reside on a curve* $\mu(\gamma)$

$$\mu(\gamma) = 2 - 2\gamma^2 + 2\sqrt{1 - \gamma^2} - 2i\gamma\big(1 + \sqrt{1 - \gamma^2}\big), \quad -\phi \le \gamma \le \phi. \tag{24}$$

Proof. Define ζ_k as

$$\zeta_k \equiv \tfrac{4+\kappa_2\lambda_{2,k}}{2\kappa_1}, \qquad k = 1, \ldots, m_2.$$

By (22) we get

$$\zeta_k = \frac{2h_1}{ch_1^\alpha} + \frac{h_1}{2h_2}\lambda_{2,k} = 2c^{-1}m_1^{\alpha-1} + w_k,$$

where

$$w_k = \tfrac{m_2}{2m_1}\lambda_{2,k}. \tag{25}$$

Lemma 5.1 together with (25) yields

$$w_k \in \mathcal{E}^+\big(2\tfrac{m_2^{\frac{1}{4}}}{m_1}, \tfrac{m_2}{m_1}(1+2m_2^{-\frac{3}{2}})\big), \qquad k = 1, \ldots, m_2.$$

By (23) $m_2 < m_1$ and $\frac{m_2}{m_1}(1+2m_2^{-\frac{3}{2}}) \leq \phi$ yielding

$$w_k \in \mathcal{E}^+(2\tfrac{m_2^{\frac{1}{4}}}{m_1}, \phi) \in \mathcal{E}^+(2m_1^{-\frac{3}{4}}, \phi).$$

Now define

$$\epsilon = c^{-1}m_1^{\alpha-1} + \delta_k m_1^{-\frac{3}{4}}, \quad 0 \leq \delta_k \leq 1,$$

which gives

$$\zeta_k = 2\epsilon + i\gamma_k, \quad -\phi \leq \gamma_k \leq \phi. \tag{26}$$

For $m_1 \gg 1$ we have $0 < \epsilon \ll 1$, and (26) combined with (18) and a Taylor expansion yields

$$z_k = i\big(\zeta_k - \sqrt{1+\zeta_k^2}\big) = 2i\epsilon - \gamma_k - i\sqrt{1-\gamma_k^2} + \frac{2\epsilon\gamma_k}{\sqrt{1-\gamma_k^2}} + \mathcal{O}(\epsilon^2).$$

We obtain

$$z_{k,\infty} \equiv \lim_{m_1\to\infty} z_k = -\gamma_k - i\sqrt{1-\gamma_k^2}, \quad -\phi \leq \gamma_k \leq \phi. \tag{27}$$

For m_1 sufficiently large there is a positive constant c_k such that

$$|z_k|^2 = 1 - \frac{4\epsilon}{\sqrt{1-\gamma_k^2}} + \mathcal{O}(\epsilon^2) < 1 - c_k m_1^{\alpha-1}.$$

Thus, we get $|z_k|^{2m_1} \to 0$, which together with (19) gives

$$\lim_{m_1\to\infty} \mu_k = 1 - z_{k,\infty}^2 + 2iz_{k,\infty}, \quad k = 1, \ldots, m_2. \tag{28}$$

By inserting (27) in (28) and exploiting the fact that $-\phi \leq \gamma_k \leq \phi$, we conclude that the eigenvalues of $M^{-1}A$ that are different from one all lie on the curve $\mu(\gamma)$ defined by (24). □

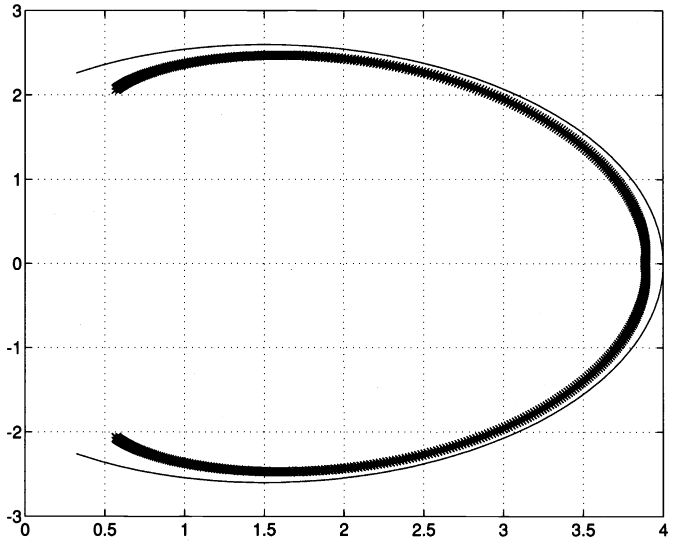


FIG. 1. *Asymptotic spectrum (solid line) and eigenvalues (stars) for $\alpha = 0.99$, $c = 100$, $\phi = 0.99$, and $m_2 = 505$.*

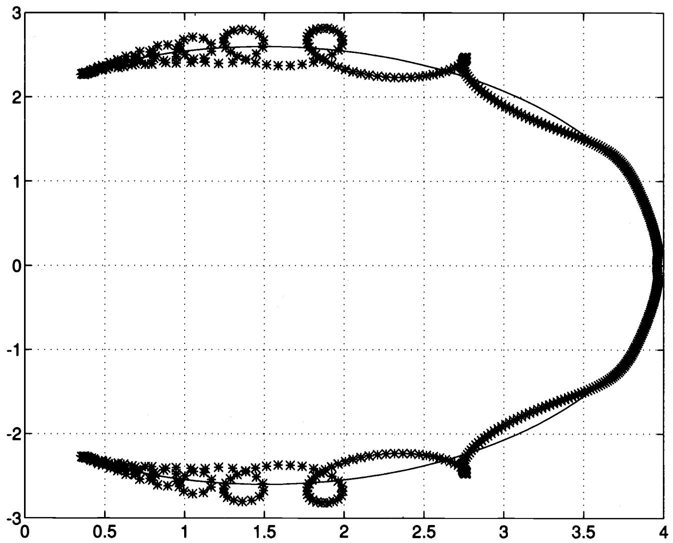


FIG. 2. *Asymptotic spectrum (solid line) and eigenvalues (stars) for $\alpha = 0.1$, $c = 10$, $\phi = 0.99$, and $m_2 = 505$.*

From Theorem 5.6 we see that when Δt is defined by (22), the eigenvalues stay bounded and are well separated from the origin as the problem size increases. If time accuracy is not required, as in steady-state computations, it is preferable to choose the time step almost independently of the space step, i.e., $\alpha \approx 0$. For time-accurate computations, the time step should be almost proportional to the space step, i.e., $\alpha \approx 1$. In Figs. 1 and 2 we show how well the asymptotic formula (24) agrees with the eigenvalues for large problems.

Numerical experiments show that $\mathrm{cond}_2(W_{M^{-1}A})$ grows rapidly with m_2. However, in Figs. 3 and 4 we note that the upper bound ψ_0 of the condition number reduction is favorable. For all values of α and the spatial grid ratio ϕ examined, ψ_0 is less than 0.1 and decreases when

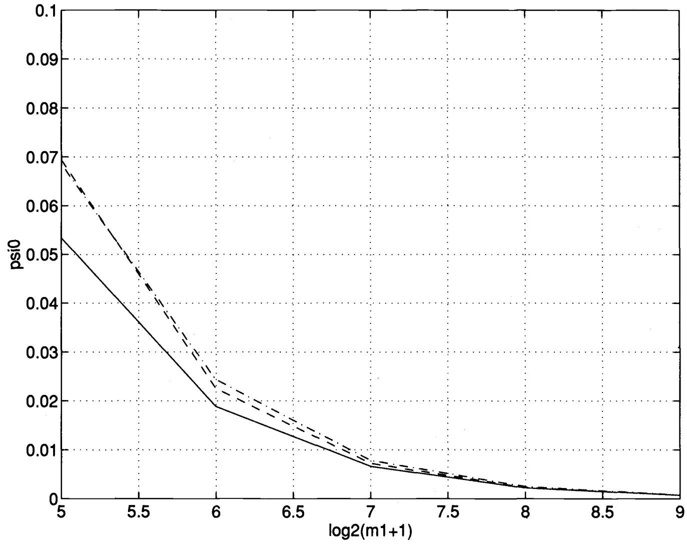


FIG. 3. *The bound* ψ_0 *for* $\alpha = 0.99$ *and* $c = 100$. *The solid line represents* $\phi = 0.99$, *the dashed line* $\phi = 0.75$, *and the dash-dot line* $\phi = 0.5$.

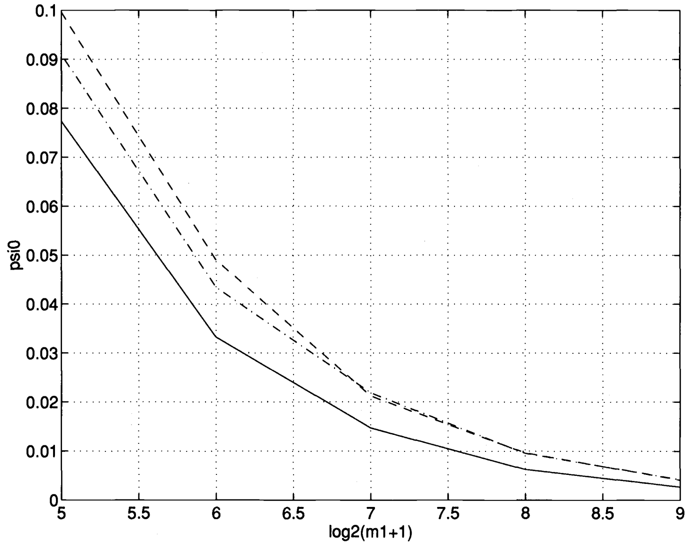


FIG. 4. *The bound* ψ_0 *for* $\alpha = 0.1$ *and* $c = 10$. *The solid line represents* $\phi = 0.99$, *the dashed line* $\phi = 0.75$, *and the dash-dot line* $\phi = 0.5$.

the problem size increases. The conclusion is that the semi-Toeplitz preconditioner improves the condition of the eigenvectors to the iteration matrix.

6. Convergence properties. In this section we determine the *asymptotic convergence factor* ρ defined by

$$\rho \equiv \lim_{i\to\infty} \varepsilon_i^{1/i}, \tag{29}$$

where ε_i is defined in (5). We enclose the asymptotic spectrum μ, defined in (24), in a circle $\mathcal{C}(4, R)$ and use the general result from [10]. Here $\mathcal{C}(c, R)$ denotes the circle with center c and radius R.

LEMMA 6.1. *The circle $\mathcal{C}(4, R)$, where R is defined by*

$$R = |\mu(\pm\phi) - 4| = \sqrt{8 + 12\phi^2 - 8\sqrt{1-\phi^2}},$$

encloses the asymptotic spectrum μ defined in (24).

Proof. Define $r(\gamma)$ as the distance between 4 and $\mu(\gamma)$. Then

$$\begin{aligned} r^2 &= \left(-2 - 2\gamma^2 + 2\sqrt{1-\gamma^2}\right)^2 + \left(2\gamma(1+\sqrt{1-\gamma^2})\right)^2 \\ &= 8 + 12\gamma^2 - 8\sqrt{1-\gamma^2} \end{aligned}$$

and

$$\frac{d^2r^2}{d\gamma^2} = 24 + \frac{8}{(1-\gamma^2)^{3/2}} > 0.$$

Hence, r^2 obtains its maximum value at the endpoints, yielding $r(\gamma) \le R$, which proves the lemma. □

THEOREM 6.2. *For $0 < \phi < \frac{\sqrt{8}}{3}$ the asymptotic convergence factor ρ satisfies*

$$\rho \le \frac{\sqrt{2 + 3\phi^2 - 2\sqrt{1-\phi^2}}}{2} < 1.$$

Proof. For $0 < \phi < \frac{\sqrt{8}}{3}$ we obtain $8 + 12\phi^2 < 8 + 12 \cdot \frac{8}{9} = 8\sqrt{1 - \frac{8}{9}} + 16 < 8\sqrt{1-\phi^2} + 16$, and consequently $\sqrt{8 + 12\phi^2 - 8\sqrt{1-\phi^2}} < 4$. Using Lemma 6.1 and the general result in [10] yields

$$\rho \le \frac{R}{4} = \frac{\sqrt{2 + 3\phi^2 - 2\sqrt{1-\phi^2}}}{2} < 1. \qquad \square$$

In Figs. 5 and 6 we display the asymptotic convergence factor as a function of ϕ. In the same figures we present the residual reduction $\tilde{\rho}$, defined by

$$\tilde{\rho} \equiv \left(\frac{\|r^{(i)}\|_2}{\|r^{(0)}\|_2}\right)^{1/i}, \tag{30}$$

for different problem sizes using generalized minimal residual (GMRES) [11] with $i = 20$.

Figs. 7 and 8 show the actual number of iterations obtained from restarted GMRES(20) for different problem sizes. The iteration has converged when $\|r^{(i)}\|_2/\|M^{-1}b\|_2$ is less than 10^{-6}.

From Figs. 5–8 we observe that the convergence seems to depend only on the spatial grid ratio ϕ and not on the number of unknowns. Finally, in Figs. 9 and 10 we present the arithmetic speedup using GMRES(20) and the semi-Toeplitz preconditioner compared to the unpreconditioned system (3). Clearly, the speedup is large for large problems, and it is also

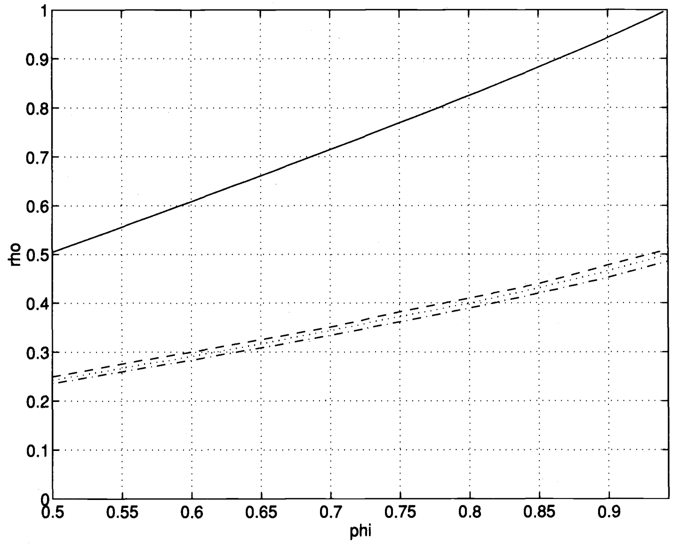


FIG. 5. *The solid line represents the asymptotic convergence factor for $\alpha = 0.99$ and $c = 100$. The residual reduction is plotted for $m_1 = 127$ (dashed), $m_1 = 255$ (dotted), and $m_1 = 511$ (dash-dot).*

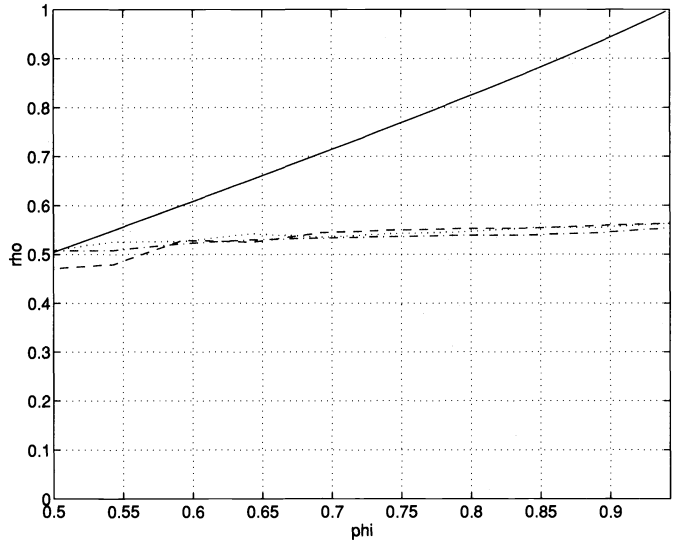


FIG. 6. *The solid line represents the asymptotic convergence factor for $\alpha = 0.1$ and $c = 10$. The residual reduction is plotted for $m_1 = 127$ (dashed), $m_1 = 255$ (dotted), and $m_1 = 511$ (dash-dot).*

greater than one for small problems. Notice that lowering ϕ yields a gain in speedup. We conclude that the semi-Toeplitz preconditioner results in a significant improvement.

7. Conclusions. We have studied semi-Toeplitz preconditioners and minimal residual iterations to solve block tridiagonal systems of equations. Analytical formulas for the eigenvalues and the eigenvectors of the preconditioned system are derived. It is numerically verified that the eigenvector matrix to the preconditioned system is better conditioned than its unpreconditioned counterpart. We have also shown that in the limit $m_d \to \infty$, the eigenvalues of the

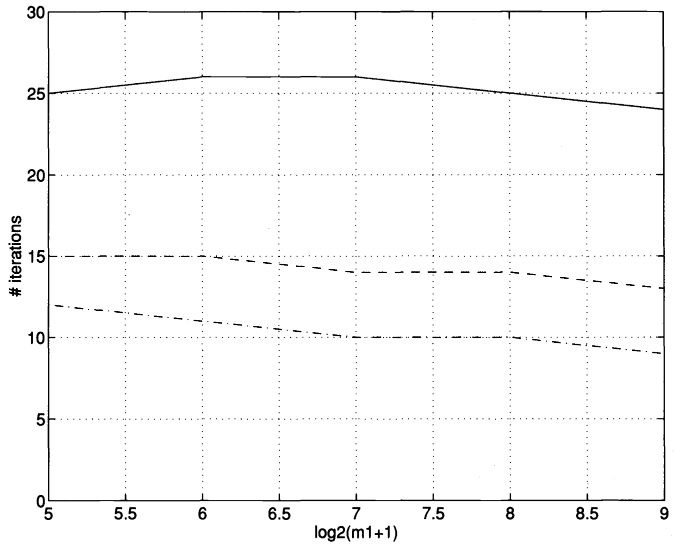


FIG. 7. *Number of iterations for* $\alpha = 0.99$ *and* $c = 100$. *The solid line represents* $\phi = 0.99$, *the dashed line* $\phi = 0.75$, *and the dash-dot line* $\phi = 0.5$.

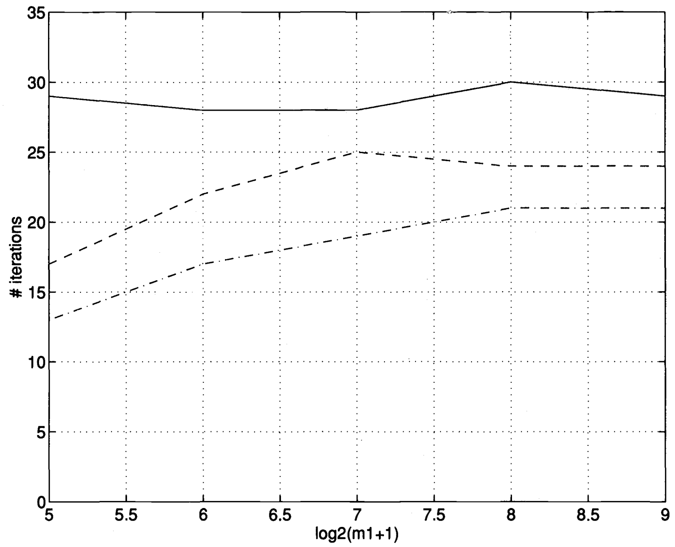


FIG. 8. *Number of iterations for* $\alpha = 0.1$ *and* $c = 10$. *The solid line represents* $\phi = 0.99$, *the dashed line* $\phi = 0.75$, *and the dash-dot line* $\phi = 0.5$.

preconditioned system all lie on a curve. The eigenvalues stay bounded and are well separated from the origin, independently of the problem size. From this eigenvalue distribution we derive upper bounds of the asymptotic convergence factor. These bounds have been corroborated numerically. Finally, we show empirically that the number of iterations does not grow when we increase the number of unknowns. This leads to a substantial arithmetic speedup compared to the unpreconditioned system.

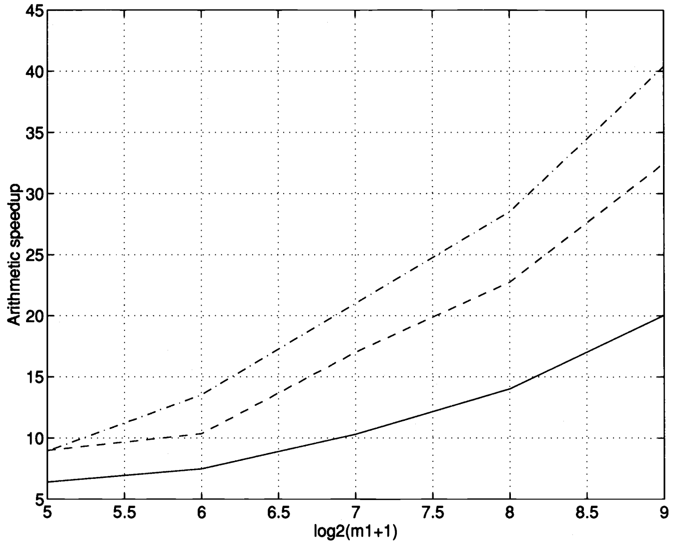


FIG. 9. *Arithmetic speedup for $\alpha = 0.99$ and $c = 100$. The solid line represents $\phi = 0.99$, the dashed line $\phi = 0.75$, and the dash-dot line $\phi = 0.5$.*

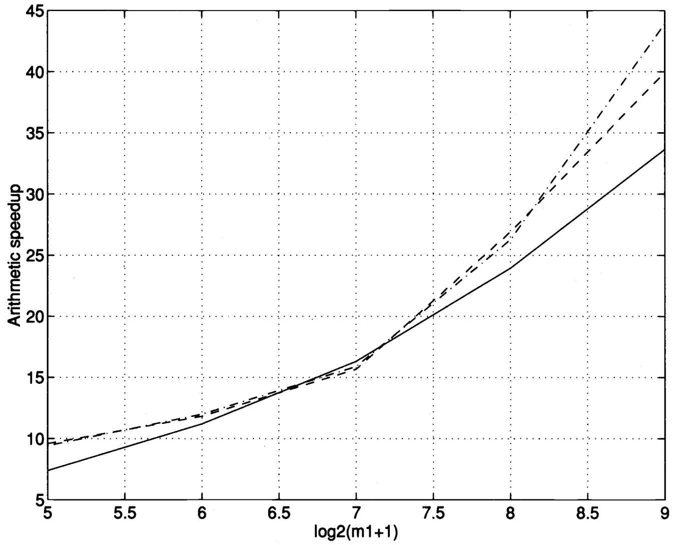


FIG. 10. *Arithmetic speedup for $\alpha = 0.1$ and $c = 10$. The solid line represents $\phi = 0.99$, the dashed line $\phi = 0.75$, and the dash-dot line $\phi = 0.5$.*

REFERENCES

[1] O. AXELSSON, *A restarted version of a generalized preconditioned conjugate gradient method*, Comm. Appl. Numer. Methods, 4 (1988), pp. 521–530.

[2] O. AXELSSON AND G. LINDSKOG, *On the eigenvalue distribution of a class of preconditioning methods*, Numer. Math., 48 (1986), pp. 479–498.

[3] R. W. FREUND, G. H. GOLUB, AND N. M. NACHTIGAL, *Iterative solution of linear systems*, Acta Numerica (1992), pp. 57–100.

[4] J. Guerra and B. Gustafsson, *A semi-implicit method for hyperbolic problems with different time-scales*, SIAM J. Numer. Anal., 23 (1986), pp. 734–749.

[5] B. Gustafsson and H. Stoor, *Navier–Stokes equations for almost incompressible flow*, SIAM J. Numer. Anal., 28 (1991), pp. 1523–1547.

[6] L. Hemmingsson, *A fast modified sine transform for solving block-tridiagonal systems with Toeplitz blocks*, Numer. Algorithms, 7 (1994), pp. 375–389.

[7] ———, *Toeplitz preconditioners with block structure for first-order PDEs*, Numer. Linear Algebra Appl., to appear.

[8] ———, *A domain decomposition method for first-order PDEs*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 1241–1267.

[9] K. Otto, *Analysis of preconditioners for hyperbolic partial differential equations*, SIAM J. Numer. Anal., 33 (1996), to appear.

[10] Y. Saad, *Krylov subspace methods for solving large unsymmetric linear systems*, Math. Comp., 37 (1981), pp. 105–126.

[11] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[12] H. A. van der Vorst, *Preconditioning by incomplete decompositions*, Ph.D. thesis, Rijksuniversiteit Utrecht, Utrecht, The Netherlands, 1982.

SIAM J. SCI. COMPUT.
Vol. 17, No. 1, pp. 65–77, January 1996

005

ODE RECURSIONS AND ITERATIVE SOLVERS FOR LINEAR EQUATIONS*

ALFRED A. LORBER†, GRAHAM F. CAREY†, AND WAYNE D. JOUBERT‡

Abstract. Timestepping to a steady-state solution is increasingly applied in engineering and scientific applications as a means for solving equilibrium problems. In the present work we examine the relation between the recursion in timestepping algorithms for semidiscrete systems of ODEs and certain types of iterative methods for solving discretized systems of equilibrium PDEs. We consider, in particular, the possibility of accelerating the ODE approach using recursions that are not time accurate together with parameter selection based on the theory of iterative methods. As one example, we take the parameters arising from the Chebyshev-type iterative methods and use them in a two-stage Runge–Kutta scheme. A comparison study for a representative steady-state diffusion problem indicates a dramatic improvement in convergence and efficiency. We remark that this approach can be trivially incorporated into existing time-integration codes to significant advantage. This yields a hybrid adaptive approach in a single code.

Key words. iterative methods, ODE solvers, Runge–Kutta, time-accurate solutions, steady-state solutions

AMS subject classifications. 65F10, 65L06

1. Introduction.

It is well known that certain simple explicit integrators may be identified with corresponding iterative methods in certain applications. For example, the forward Euler integration scheme for unsteady diffusion can be easily related to the point Jacobi iteration for the steady-state problem. The purpose of this study is twofold: first to investigate the connection between the solution of systems of linear equations by iterative methods and the steady-state integration of a related class of evolution PDEs by timestepping methods, and second to exploit this relationship to accelerate the ODE approach by improved parameter selection. This then permits an adaptive hybrid strategy in which either time accurate solutions or steady solutions can be computed efficiently by changing parameters in a single algorithm and code.

In engineering applications ODE integrators are increasingly being used to timestep a linear or nonlinear ODE system to steady state. This may be done in two ways: values of the solution may be obtained for intermediate time values that correspond to the actual physical solution at those intermediate times (time-accurate); alternatively, intermediate solution approximations may be obtained that have no relation to the physical solution at actual intermediate times (non-time-accurate). This strategy is not only convenient in the sense that the same algorithm or code is used for both classes of problems, but, particularly for nonlinear problems, the time-iterative algorithms may exploit the physics of the evolution problem to produce a more robust algorithm [1].

The preceding remarks notwithstanding, it is more common to obtain the steady-state solution directly using a linear system solver. Iterative solvers are particularly attractive for large-scale systems arising from discretized PDEs. The iterates produced by such solvers can be compared with intermediate solution approximations produced by the ODE recursion. Of course, usually the iterates do not correspond to time-accurate solutions, though they may for some iterative methods and problem classes. For nonlinear ODE problems, iterative linear solvers may be used to solve linearizations of the ODE problem at incremental steps toward the steady-state solution (see, e.g., [3]). Similarly, continuation techniques may be applied to

*Received by the editors June 3, 1994; accepted for publication (in revised form) January 26, 1995. This work was supported in part by ARPA through grant DABT63-92-C-0024, by the Department of Energy through grant DE-FG03-93ER25183, and by Texas ATP, with the University of Texas at Austin. This work was also supported in part by the Department of Energy through grant W-7405-ENG-36, with Los Alamos National Laboratory.

†CFD Lab, WRW 111, The University of Texas at Austin, Austin, TX 78712 (aal@cfdlab.ae.utexas.edu).

‡Los Alamos National Laboratory, Los Alamos, NM 87545 (wdj@lanl.gov).

general nonlinear (or linear) problems by embedding them within a nonlinear ODE problem and applying ODE solvers. In such cases, the continuation parameter can be interpreted as a "time-like variable" for the associated nonlinear ODE.

The theory of ODE solvers has mainly focused on issues of accuracy (with respect to time), stiffness, and stability. The interest in timestepping the transient problem to get a steady-state solution is relatively little studied in its own right, but is being increasingly used (particularly in engineering computations for nonlinear problems with complex physics where different rate processes may occur and this feature can be exploited [5]). On the other hand, there is an extensive theoretical foundation on iterative methods (see, e.g., [17], [18], [20]). This includes analysis of convergence behavior, stopping tests, and other properties of the various iterative methods. More importantly, in the present context, it provides an analysis of parameter selection for "parameter-dependent" methods such as Chebyshev and for adaptively selecting relaxation factors. These analytical results can be exploited to derive improved ODE recursions as shown later. Moreover, an existing ODE scheme can be easily modified to a hybrid form that permits faster steady-state solutions through appropriate parameter selection. These ideas are similar to those used with ODE exponential fitting methods. This also implies that in a given calculation the parameters can be adapted so that, for instance, the early transient is computed time-accurately, then the steady state is computed using a local time-inaccurate scheme with appropriate parameters, and finally the transient "off-design" behavior can be examined time-accurately.

The outline of the present paper is as follows: in §2 we consider a simple class of linear evolution equations and give the exact integral solution involving the exponential matrix. This motivates the discussion of time-accurate and steady-state behavior. The problem of integrating semidiscrete ODE systems of this form is then introduced and several standard ODE methods are listed together with a brief discussion of approximations of the exponential. Using these properties, the relationship between the ODE recursion and iterative schemes is examined in §§4 and 5. In particular, we list here a number of different iterative methods and identify corresponding ODE recursions. The special case of polynomial preconditioners and Chebyshev-type methods follow in §6. A modified Runge–Kutta scheme is then developed using the Chebyshev iterative parameters and applied to a diffusion problem in §§7 and 8. This demonstrates the main ideas and indicates the value of this new hybrid approach.

2. Evolution equations. To motivate the approach let us begin with the simple time-dependent problem

$$u'(t) = f(t) - A(t)(u(t)) \equiv R(t, u(t)), \quad t \geq 0, \qquad u(0) = u_0. \tag{1}$$

Here, $u : \mathbb{R}^+ \to V$, and $A : \mathbb{R}^+ \times V \to V$ is a linear or nonlinear operator for some linear space V (see [14]). When f and A are independent of t, then the steady-state solution of (1) is defined by $u' = R(\cdot, u) = 0$.

In the case when A is linear and satisfies certain standard conditions, the solution to (1) is given uniquely by

$$u(t) = e^{-At}u_0 + \int_0^t e^{-As} f(t-s)ds. \tag{2}$$

If furthermore f is independent of time and the integration is successful, then the solution can be simply expressed as

$$u(t) \quad = \quad u_0 + A^{-1}[I - e^{-At}](f - Au_0). \tag{3}$$

If we introduce the residual $r(t) \equiv R(t, u(t))$ in (1), then (3) can be rewritten equivalently as

$$u(t) \quad = \quad u_0 + A^{-1}[I - e^{-At}]r(0), \tag{4}$$

which implies

$$r(t) = e^{-At} r(0). \tag{5}$$

For this case, when A is nonsingular with eigenvalues having positive real part, then as $t \to \infty$, $e^{-At} \to 0$. Thus $u(t) \to u_0 + A^{-1} r(0) = u_0 + e^{(0)} = u_\infty$, where u_∞ denotes the solution to the steady-state problem and $e^{(0)}$ denotes the initial error $e^{(0)} = u_0 - u_\infty$ measured relative to the steady-state solution. Furthermore, setting $t = t_n$ and $\Delta t_n = t_n - t_{n-1}$ for a timestep in (3) and (5), we obtain

$$u(t_n) = u(t_{n-1}) + A^{-1}[I - e^{-A\Delta t_n}](f - Au(t_{n-1})) \tag{6}$$

and

$$r(t_n) = e^{-A\Delta t_n} r(t_{n-1}). \tag{7}$$

Hence a discrete set of values $\{u(t_n)\}$ can be computed using (6). As $n \to \infty$ the sequence converges to the steady-state solution u_∞. Similarly, the residual sequence $\{r(t_n)\}$ converges to zero. For illustrative purposes we have selected a problem that can be integrated exactly (although in computation the exponential operator would still need to be evaluated, possibly by a Padé approximation), and A^{-1} appears on the right side. This result serves to motivate our approach.

Thus, the incremental time-stepping recursion (6) may be viewed as a form of iterative method that converges to the steady-state solution $u_\infty = A^{-1} f$ as $n \to \infty$. Obviously, the process is "time exact" in the sense that computed intermediate values $\{u(t_n)\}$ are precisely the transient solution of the evolution problem at corresponding times $\{t_n\}$. In practice, the system (1) is integrated approximately in which case the ODE integrator generates a sequence of approximations u_n to $u(t_n)$ with corresponding residuals r_n. If the timestep Δt_n is sufficiently small, then $u_n \sim u(t_n)$ with $u(t_n) = u_n + O((\Delta t)^p)$ where p is the order ($p = 1$ for a first-order-in-time scheme, etc.). Different numerical schemes may be used and these would lead to different classes of recursions for an iteration to compute an approximation to u_∞. Of particular interest is the case where V is finite dimensional, so A is a matrix. This is frequently the case for approximate solution of PDEs.

The convergence of these integration schemes toward u_∞ may be improved by relaxing the requirement of time-accuracy and increasing the size of the timestep. Certain stability conditions on the timestep still may apply, but even these may be "occasionally" violated during the recursion without destroying convergence. In fact, different local timesteps may be taken for the components of a vector $u(t)$. The method is no longer an ODE integrator in the usual sense but can still be viewed as an iteration for u_∞. Standard iterative methods for solving systems are not developed in this way. Nevertheless, the approach of time-stepping to a steady state and related ideas in continuation theory are extensively used in engineering practice with considerable success. Part of the focus of the present work is the relationship between the corresponding iterative recursions obtained by these approaches. From this relationship we show by means of an example with RK integration how iteration theory can be exploited to obtain faster convergence from the time-stepping recursion.

3. ODE solvers. Let us now consider some specific classes of methods for solving (1). We construct approximations $\{u_i\}$ of u at the times $\{t_i\}$, where $0 = t_0 < t_1 \cdots < t_n \cdots \leq \infty$. Standard ODE solvers attempt to determine u_i by using approximate values of u at previous times, as well as evaluations of $R(\cdot, \cdot)$. ODE solvers may be classified as follows (e.g., [19]).

First, set $\Delta t_i = \Delta t$ as a constant "parameter." The linear multistep methods are defined by

$$\sum_{i=0}^{k_1} \alpha_i u_{n-i} = \Delta t \sum_{i=0}^{k_2} \beta_i R(t_{n-i}, u_{n-i}) \tag{8}$$

for some k_1, k_2, and for $\alpha_0 \neq 0$. The method is an explicit method if and only if $\beta_0 = 0$. This method can be easily generalized to cases where not all Δt_i are the same. If all α_i are zero except α_0 and one other, then numerical quadrature rules may be used in order to construct a specific method, such as the forward Euler scheme $u_n = u_{n-1} + (\Delta t) r_{n-1}$ or, in terms of residuals, $r_n = (I - \Delta t A) r_{n-1}$. On the other hand, if only one β_i is nonzero, numerical differentiation formulas may be employed. Note that in order for the method to be exact when u is (pointwise) constant independent of time, we must have $\sum \alpha_i = 0$.

A second well known class of methods, which we consider in more detail later, is the Runge–Kutta (RK) family, which can be expressed in the form

$$u_n = u_{n-1} + \sum_{i=1}^{s} b_i \kappa_i, \tag{9a}$$

with

$$\kappa_i = \Delta t R \left(t_{n-1} + c_i \Delta t, u_{n-1} + \sum_{j=1}^{i-1} a_{ij} \kappa_j \right), \tag{9b}$$

where $c_i = \sum_{j=1}^{i-1} a_{ij}$ ($c_1 = 0$), and s is the number of stages in the method. The coefficients b_i and a_{ij} are chosen so that the coefficients of the increasing powers of Δt in a Taylor series expansion of the right- and left-hand sides of (9a) agree exactly up to a desired order. Under reasonable assumptions, this means that a certain order of accuracy is obtained as $\Delta t \downarrow 0$. Other properties are of interest as well, such as the behavior of the method if Δt is too large: a method is said to be unconditionally stable if $u_i \to u_\infty$ as $i \to \infty$, for any u_0, regardless of the size of Δt.

Now let us briefly review some familiar forms of these methods for the case where f and A are constant (for later convenient comparison with corresponding iterative methods, we express these directly in terms of the residuals):[1]

$$\begin{array}{ll}
\text{forward Euler} & r_n = [I - (\Delta t A)] r_{n-1}, \\
\text{backward Euler} & r_n = [I + (\Delta t A)]^{-1} r_{n-1}, \\
\text{Crank–Nicolson} & r_n = [I - (\frac{\Delta t}{2} A)][I + (\frac{\Delta t}{2} A)]^{-1} r_{n-1}, \\
\text{Runge–Kutta second order} & r_n = [I - (\Delta t A) + \frac{1}{2}(\Delta t A)^2] r_{n-1}, \\
\text{Runge–Kutta third order} & r_n = [I - (\Delta t A) + \frac{1}{2}(\Delta t A)^2 - \frac{1}{6}(\Delta t A)^3] r_{n-1}, \\
\text{Runge–Kutta fourth order} & r_n = [I - (\Delta t A) + \frac{1}{2}(\Delta t A)^2 - \frac{1}{6}(\Delta t A)^3 \\
 & \quad + \frac{1}{24}(\Delta t A)^4] r_{n-1}.
\end{array} \tag{10}$$

It is common to investigate these schemes from the standpoint of stability: if $r_n = g(\Delta t A) r_{n-1}$, then $r_n \to 0$ as $n \to \infty$ is assured if $|g(\Delta t \lambda_i)| < 1$ for each eigenvalue $\lambda_i \in \sigma(A)$. This analysis is valid as long as g is a rational function. Instead of beginning

[1] Here, we give RK methods in which the order p is equal to the number of stages s. In general, $p \leq s$.

with a method and then determining where $|g| < 1$, an alternate approach is to use the spectral properties of A to determine better coefficients for the method. This approach will be considered later in this study.

Each of the above examples in (10) can be identified as an approximation of the form $r_n \doteq e^{-\Delta t A} r_{n-1}$ for (7). Specifically, we seek approximations of the exponential of the form

$$e^{-A\Delta t} \doteq I - AG_{\Delta t} \tag{11}$$

(or $G_{\Delta t} \doteq A^{-1}[I - e^{-A\Delta t}]$) for some computable operator $G_{\Delta t}$. Then

$$r_n = [I - AG_{\Delta t}]r_{n-1} \tag{12a}$$

or

$$u_n = u_{n-1} + G_{\Delta t} r_{n-1}. \tag{12b}$$

These ideas can also be related to the Padé approximations of the exponential by rational functions. Similar concepts arise in semigroup theory (e.g., see [14]).

The backward Euler and Crank–Nicolson schemes in (10) are implicit and unconditionally stable for all Δt: in this case we can consider $\Delta t \to \infty$ in (11) and (12) so that $G_{\Delta t}$ then approximates A^{-1}. The other schemes are explicit and there is a maximum possible stepsize Δt for stability. Usually, the approximation $G_{\Delta t}$ is constructed to give the highest possible accuracy in the sense of asymptotic rate with respect to Δt, and within a given class of methods. However, the size of the stability region will also vary with the $G_{\Delta t}$ form chosen.

4. Iterative solvers and ODE integrators. Iterative linear solvers involve recursions and hence have a similar structure to the ODE recursions above. When solving linear equations directly by iterative methods, an approximation of zero in (11) is sought instead of an approximation to the exponential. That is, for linear iterative solvers an operator F_n analogous to $G_{\Delta t}$ in (12) is needed such that the approximation $I - AF_n \doteq 0$ holds in some sense. In this case, since $r_n = [I - AF_n]r_{n-1}$ and $\|r_n\| \le \|I - AF_n\| \cdot \|r_{n-1}\|$, making $\|I - AF_n\|$ small in turn makes $\|r_n\|$ small. We then obtain an iteration process defined by

$$u_n = u_{n-1} + F_n r_{n-1}, \tag{13}$$

which can be compared with (12). An alternate formulation is obtained by writing $u_n = u_{n-1} + K_n c_n$, where the columns of K_n are a basis for some space and c_n is chosen so that u_n is as close to u as possible [8].

It is clearly desirable that F_n be an approximation of A^{-1} in some sense. The most simple choice is $F_n = I$, leading to the *basic iterative method*

$$u_n = u_{n-1} + r_{n-1}. \tag{14}$$

Clearly, the more A departs from the identity, the worse the choice of $F_n = I$, so while this choice is simple and obviously inexpensive, we anticipate that convergence of (14) may be slow. Other choices of F_n that lead to faster iterative schemes are possible, leading to various combinations of preconditioners with iterative acceleration techniques. Examples of these will be discussed in the next section. In fact, an important issue is the selection of parameters to accelerate convergence, and this is central to the current work. In fact, drawing a relationship between ODE solvers in (12) and iterative solvers in (13) immediately suggests two avenues for the development of improved methods. First, ODE solvers may be modified to give the steady-state solution faster by abandoning time-accuracy and making the best choice of parameters for accelerating convergence to the steady state rather than embracing accuracy. Second, iterative linear solvers may be modified to give time-accurate intermediate solutions by modifying the given solver to approximate the exponential instead of approximating zero. In each case, the result is a "hybrid" solver with software that can provide both time-accurate evolution solutions and efficient steady-state solutions.

When A is linear and independent of time, a linear solver may be preferable for determining the steady-state solution. However, when A is nonlinear or dependent on t, it may be more desirable to use ODE recursions to get the steady-state solution, since the predictability of the solution based on a linearization of A at the current time is more tenuous. In fact, in some applications an artificial transient term is added to convert the problem to a time-dependent ODE form that can then be integrated to steady state.

5. Polynomial methods and recursion equivalence. In this section we describe some specific relationships between iterative linear solvers for matrix A and ODE solvers that lead to an approach for parameter selection. This discussion is loosely based on the summary of iterative linear solvers in [8]. We remark, however, that our analysis applies to more general operators A.

An effective choice of the operators F or G is given by a polynomial in the matrix A. Linear solvers of this class are known as *polynomial* or *Krylov subspace methods.* A generalization of this is the class of *preconditioned polynomial methods*: for some matrix Q that approximates A, we let $F = q_{n-1}(Q^{-1}A)Q^{-1}$. Likewise, left, right, or two-sided preconditioning could be considered by using $F = Q_R^{-1}q_{n-1}(Q_L^{-1}AQ_R^{-1})Q_L^{-1}$ for appropriate Q_L and Q_R where $Q_L^{-1}AQ_R^{-1} \doteq I$.

The most fundamental polynomial method for solving linear equations is the extrapolated basic iterative method, for which $F = \alpha I$, for some α. Typically, α is chosen to minimize the spectral radius of $I - AF = I - \alpha A$. The recursion for this iterative linear solver is identical to that for Euler's ODE scheme, where $F = G$ with the timestep given by $\Delta t = \alpha$. The method with $\alpha = 1$ is simply referred to as the (unextrapolated) basic iterative method.

The extrapolated basic iterative method may be coupled with preconditioning by writing $F = \alpha Q^{-1}$. Of course, the resulting iterative method does not generally now yield time-accurate solutions. It can be interpreted as yielding time-accurate solutions with Euler's method for a modified ODE where the operator A is replaced by the operator AQ^{-1} with f replaced by $Q^{-1}f$ and again $\Delta t = \alpha$. For example, let us decompose A into diagonal and strictly lower and upper triangular parts as $A = A_D + A_L + A_U$, and set $Q = A_D$. For $\alpha = 1$, Jacobi's method for solving linear equations is obtained. When α is varied from 1, then a relaxation form of the Jacobi iteration is obtained. More generally, if $Q = \Omega$ where Ω is diagonal, the iterative method can be interpreted as corresponding to the use of a modified Euler method in which different timesteps are used at different points in the physical domain.

When the basic iterative method is applied with a preconditioning of $Q = \frac{1}{\omega}A_D + A_L$, then the SOR method is obtained, with the special case of $\omega = 1$ being the Gauss–Seidel method. As pointed out in [17], this method also is equivalent to a first-order, time-accurate method for the operator $A_D^{-1}A$, where ω is interpreted as the timestep and is chosen to be near zero. The ad hoc SOR method [4] employs different values of ω at different grid points in the mesh. This can be expressed in the form $Q = \Omega^{-1}A_D + A_L$ where Ω is diagonal and can also be seen to give time-accurate solutions for the operator $(\Omega/\Delta t)A_D^{-1}A$, where Δt is the timestep. Conversely, this scheme can be interpreted as an ODE-type method that is not time-accurate for operator A and that uses varying timesteps for different components; i.e., for the PDE problem this would correspond to local timesteps at different parts of the physical domain. Similarly, chaotic relaxation techniques [2] may be viewed as an extreme form of

ODE scheme that applies timesteps at various locations in the physical domain according to some ordering, yielding solutions, which, in general, are not time-accurate.

The SSOR preconditioner $Q = \frac{\omega}{2-\omega}(\frac{1}{\omega}A_D + A_L)A_D^{-1}(\frac{1}{\omega}A_D - A_U)$ used with the basic iterative method also corresponds to a first-order, time-accurate method applied to operator $A_D^{-1}A$, using timestep $\omega(2-\omega)$; again, ω must be chosen near zero to give good time-accuracy. Furthermore, when A is a property-A matrix [18], the incomplete Cholesky and modified incomplete Cholesky preconditioners have the same form for Q, except that A_D is replaced with a diagonal matrix of the pivots; this preconditioner with the basic iterative method may again be interpreted equivalently as an ODE solver that uses different timesteps over a spatial discretization of a physical domain.

Other preconditioners can be interpreted similarly in the context of ODE solvers; for example, [17] describes the relationship between the Peaceman–Rachford preconditioner and the Crank–Nicolson method. Similarly, the ADI solvers may be related to operator-split ODE integration schemes. Finally, multigrid methods may be used as linear solvers by seeking appropriate multigrid approximations of zero, while the work by Jesperson [7] may be seen as an attempt to form multigrid approximations to the exponential. Pardhanani [11] has developed similar multigrid schemes for semidiscrete ODE systems.

6. Polynomial methods to enhance ODE solvers. The purpose of this section is to explore new ODE-type methods that may be derived using known results on polynomial approximations. Following [8], polynomial methods for solving linear equations may be divided into two categories: *Chebyshev-type methods*, for which the polynomial q_{n-1} is independent of r_0 and is usually based on the theory of polynomials in the complex plane, and *conjugate gradient-type methods*, for which q_{n-1} is based on r_0, typically through inner-product information. A third class of *hybrid* or *adaptive* methods mixes the approaches, usually to supplement the Chebyshev-type method with spectral estimates for A from the conjugate gradient component of the algorithm.

Chebyshev-type methods generally attempt to make $P_n(z) = 1 - zq_{n-1}(z)$, for z complex, approximate zero on some $\Sigma \supseteq \sigma(A)$ (the spectrum of A), or alternatively on its boundary $\partial\Sigma$. The approximation is usually done either in the ℓ^∞ norm (preferable) or the ℓ^2 norm. A wide range of methods has been developed for this problem. Of particular interest are those for which Σ is an ellipse. Here, if an ellipse containing $\sigma(A)$ can be described, a corresponding $P_n(z)$ may be determined that is parameterized by variables that describe the ellipse. This is done in §7.

Similarly, many Chebyshev-type methods may be modified immediately to approximate e^{-tz} instead of zero on Σ or $\partial\Sigma$, yielding new methods for obtaining time-accurate ODE solutions. In particular, we may minimize $\sup_{z\in\Sigma}|e^{-\Delta tz} - (I - zF(z))|$, where F is a polynomial and $\sigma(A) \subseteq \Sigma \subseteq \mathbb{C} \setminus \{0\}$. This approach corresponds to constructing iterative linear equation solvers of the Chebyshev type for which $\sup_{z\in\Sigma}|(I - zF(z))|$ is minimized. This technique has the advantage over conjugate gradient-type methods in that it requires fewer inner products, which may be advantageous on certain parallel computers. A more general Chebyshev-type approach is to minimize some other norm of the function $e^{-\Delta tz} - (I - zF(z))$ over Σ, such as a least-squares norm.

On the other hand, conjugate gradient-type methods for solving $Au = f$ can be viewed as a special case of *projection methods*, defined by the relations

$$u_n \in u_0 + \mathbf{R}_n, \qquad e_n = u_n - u \perp \mathbf{L}_n \tag{15}$$

for some linear spaces $\mathbf{L}_n$ and $\mathbf{R}_n$. When $\mathbf{L}_n = B\mathbf{R}_n$ for some matrix B and, furthermore, if B is Hermitian positive definite (HPD), then the orthogonality condition causes $\|e_n\|_B$ to be minimized, where $\|v\|_B = \sqrt{v^*Bv}$. To find time-accurate solutions to the ODE problem,

(15) may be generalized to

$$u_n \in u_0 + \mathbf{R}_n, \qquad u_n - u(t) = -A^{-1}[r_n - e^{-At}r_0] \perp \mathbf{L}_n. \tag{16}$$

Again, if $\mathbf{L}_n = B\mathbf{R}_n$ for HPD B, then $\|A^{-1}[r_n - r(t)]\|_B$ is minimized.

Letting L_n and R_n be matrices whose columns form a basis for $\mathbf{L}_n$ and $\mathbf{R}_n$, respectively, we obtain from (16) and (4)

$$u_n = u_0 + R_n[L_n^* R_n]^{-1} L_n^* A^{-1}[I - e^{-At}]r_0. \tag{17}$$

Unfortunately, the quantity $L_n^* A^{-1} e^{-At} r_0$ typically is not known; thus, it must be approximated, e.g., by replacing A in e^{-At} with a projection of A available from the method.

When the method is a polynomial method, $\mathbf{R}_n = K_n(r_0, A) \equiv \operatorname{span}\{A^i r_0\}_{i=0}^{n-1}$, which is the Krylov space. Two important choices of $\mathbf{L}_n$ are $\mathbf{L}_n = A^* A K_n(r_0, A)$, which gives rise to the GMRES linear solver and related methods, and $\mathbf{L}_n = A^* K_n(\tilde{r}_0, A^*)$, $(\tilde{r}_0 = r_0)$, which gives rise to the biconjugate gradient method and related methods.

We now derive a time-accurate ODE solver based on the biconjugate gradient approach. The biconjugate gradient method generates matrices R_n and $\tilde{R}_n \equiv A^{-*} L_n$ whose columns form bases for $K_n(r_0, A)$ and $K_n(\tilde{r}_0, A^*)$, respectively, such that $\tilde{R}_n^* A R_n = D_n$ is diagonal and $\tilde{R}_n^* A^2 R_n = T_n$ is tridiagonal. Let us assume that $R_n = [\, R_{n-1} \quad v_n \,]$ and $\tilde{R}_n = \left[\, \tilde{R}_{n-1} \quad \tilde{v}_n \,\right]$ for each n. Using the approximation $D_n^{-1} \tilde{R}_n^* A(A)^i R_n \doteq (D_n^{-1} T_n)^i$, $i \geq 0$ (which is exact when $i < 2$ or when R_n and $\tilde{R}_n$ are square and nonsingular), we obtain

$$\begin{aligned} u_n &= u_0 + R_n D_n^{-1} \tilde{R}_n^* A [z^{-1} - z^{-1} e^{-zt}]_{z=A} R_n \beta e_1 \\ &\doteq u_0 + R_n [z^{-1} - z^{-1} e^{-zt}]_{z=D_n^{-1} T_n} \beta e_1 \end{aligned} \tag{18}$$

for some $\beta \neq 0$, since $R_n e_1$ is proportional to r_0. This algorithm may be implemented by evaluating the analytic function $[z^{-1} - z^{-1} e^{-zt}]$ applied to the matrix $D_n^{-1} T_n$ using, for example, a Jordan decomposition of the matrix. The result is a biconjugate gradient-type ODE solver for obtaining time-accurate solutions. Similarly, in [6] a time-accurate ODE solver is developed based on the GMRES approach.

It should be noted that these different approaches, whether Chebyshev-type or CG-type, can be modified so that the polynomials generated yield higher-order ODE solvers. This can be done by putting constraints on the derivatives of the polynomial P_n at zero. The inverse problem of adapting a time-accurate ODE integrator to an improved iterative recursion may also be addressed. In the next section we consider this approach for the RK ODE schemes and then apply the resulting scheme in a numerical experiment.

7. A modified RK scheme. In this section we construct a RK ODE recursion based on applying ideas from iterative linear system solvers to the problem of finding steady-state solutions to a (possibly nonlinear) ODE. Using the notation introduced in (10), the two-stage RK methods for constants A and f can be conveniently expressed as the residual recursion

$$r_n = g(\Delta t A) r_{n-1}, \tag{19}$$

where

$$g(\Delta t A) = I - \gamma_1(\Delta t A) + \gamma_2(\Delta t A)^2. \tag{20}$$

Comparing (19) and (20) with (9) for $s = 2$, we have $\gamma_1 = b_1 + b_2$ and $\gamma_2 / b_2 = c_2 = a_{21}$ (see also [13]).

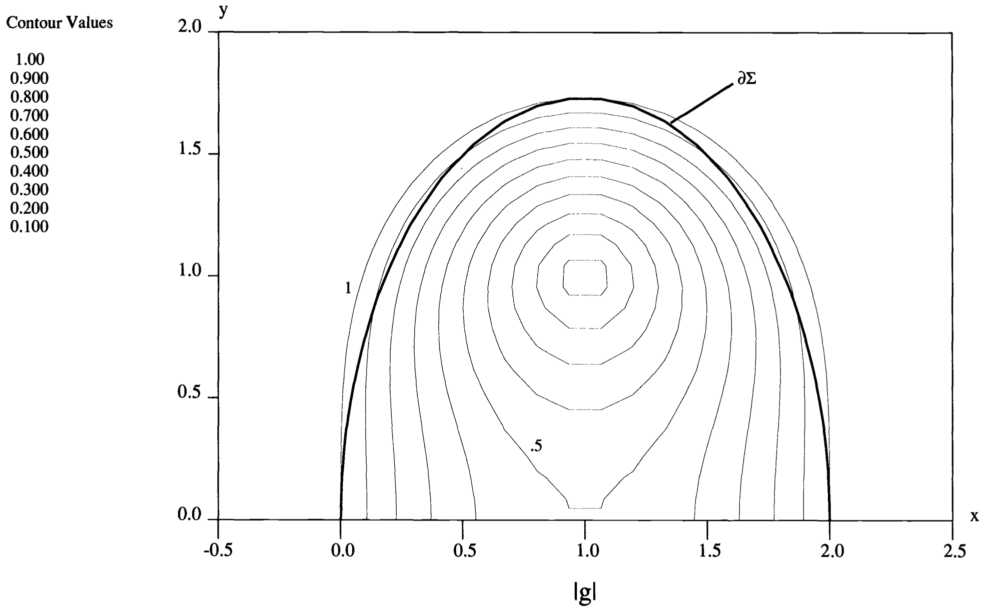


FIG. 1. *Stability region: second-order, two-stage RK, and the associated boundary of (elliptical region)* Σ.

Second-order accuracy is obtained when γ_1 and γ_2 are set equal to 1 and $\frac{1}{2}$, respectively, and the RK2 scheme in (10) is recovered. If instead we set only $\gamma_1 = 1$ and allow γ_2 to be free, the resulting scheme is only first-order accurate. However, γ_2 can now be chosen to enhance convergence to the steady state and thereby create an improved iterative recursion. The idea extends immediately to higher-order RK formulas and to other methods.

For a method of given order, the size of the timestep, and hence the effectiveness of an ODE integrator, can be measured by analyzing the stability domain of the method. In §3, we saw that stability is assured if $|g(\Delta t\lambda_i)| < 1$ for each eigenvalue $\lambda_i \in \sigma(A)$. That is, the stability domain corresponds to the region in the complex plane $(\mathrm{Re}\,(\Delta t\lambda)\,, i\mathrm{Im}\,(\Delta t\lambda))$ for which $|g(\Delta t\lambda)| < 1$.

The stability domain for the second-order scheme above ($\gamma_1 = 1$, $\gamma_2 = \frac{1}{2}$) is shown in Fig. 1. Here, we have plotted the contours $|g(\zeta)| = .1, .2, \ldots, 1$ in the upper-half complex plane. If instead only γ_1 is specified, then γ_2 can be chosen to increase the size of the stability domain. This, in turn, may lead to a larger stepsize and accelerate "convergence" of the recursion to the steady state. Moreover, in view of the correspondence noted in §5, we can appeal to parametric selection criteria from the theory of iterative methods to determine an optimal value of γ_2.

Here, we investigate selecting values for γ_2 based on the properties of Chebyshev-type polynomial iterative methods discussed in §6. These methods were seen to maximize residual reduction per iteration on $\Sigma \supseteq \sigma(A)$ as measured by a suitable norm. To produce iteration polynomials $P_n(z)$ with characteristics similar to those of the RK methods being considered, we choose Chebyshev-based iteration polynomials for which Σ is an ellipse passing through the origin and centered at coordinate $(d, 0)$ with foci at $(d \pm c, 0)$. These iteration polynomials are given by

$$P_n(z) = \frac{T_n\left(\frac{z-d}{c}\right)}{T_n\left(\frac{-d}{c}\right)}, \tag{21}$$

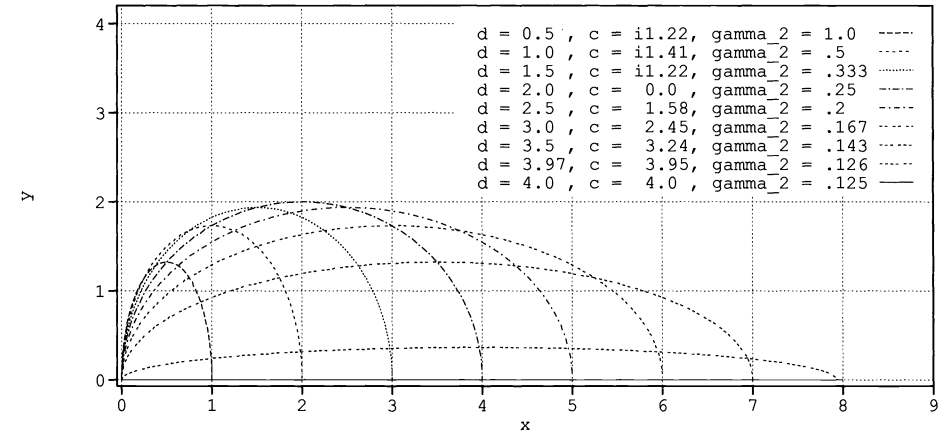


FIG. 2. *Boundaries of elliptical domains* Σ *and corresponding* γ_2 *for representative values of* d *and* c.

where c can be either pure real or pure imaginary [10]. Here n is the degree of the polynomial and T_n are Chebyshev polynomials of the first kind. For the two-stage RK example considered here, the quadratic Chebyshev polynomial is

$$T_2(z) = -1 + 2z^2, \tag{22}$$

so that in (21)

$$P_2(z) = 1 - \frac{4dz}{-c^2 + 2d^2} + \frac{2z^2}{-c^2 + 2d^2}. \tag{23}$$

Since we want the polynomial g in (20) to correspond to the iteration polynomial in (23), this implies

$$\gamma_1 = \frac{4d}{-c^2 + 2d^2} \quad \text{and} \quad \gamma_2 = \frac{2}{-c^2 + 2d^2}. \tag{24}$$

For instance, for the second-order scheme (10) with $\gamma_1 = 1$ and $\gamma_2 = \frac{1}{2}$ in (24),

$$d = 1, \qquad c = \pm i\sqrt{2}. \tag{25}$$

Obviously, substituting for c and d in (23) and sketching the level contours of $P_2(z)$ again yields the plot in Fig. 1. Note that as expected, the stability domain contains the elliptical domain Σ with center and foci defined by (25) and sketched in Fig. 1.

If, instead, the desired accuracy is relaxed to first order by setting $\gamma_1 = 1$, then (24) generates an alternative method for selecting γ_2. We then have

$$c = \pm\sqrt{2d\,(d-2)} \quad \text{and} \quad \gamma_2 = \frac{1}{2d}, \tag{26}$$

where d may be chosen based on knowledge of $\sigma(A)$. A set of ellipses corresponding to representative values of d and c are sketched in Fig. 2. The ellipses are defined for values of d between 0 and 4 and, in general, as d increases the ellipses become increasingly elongated in the positive real direction. At $d = 0$, $c = 0$ and the ellipse degenerates to a point. For $0 < d < 2$, c is imaginary and the semimajor axes are parallel to the imaginary axis. At

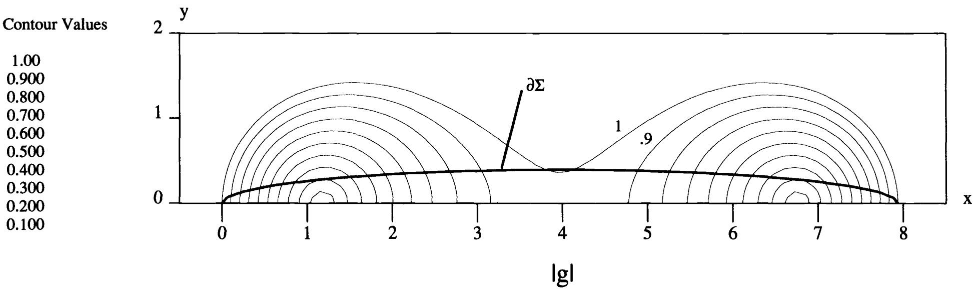


FIG. 3. *Stability region: first-order, two-stage RK* ($\gamma_1 = 1.0, \gamma_2 = .1260$), *and the associated boundary of* (*elliptical region*) Σ.

$d = 2$, $c = 0$ and the ellipse is a circle. For $2 \leq d < 4$, c is real and the semimajor axes are parallel to the real axis. At $d = c = 4$, the ellipse degenerates to a slit of length 8. Increasing d beyond 4 generates γ_2 values for which the stability regions have a detached lobe. Hence d (and therefore γ_2) can be selected to accelerate convergence for a given class of problems. This is illustrated in the next section for a representative diffusion problem. Since the eigenvalues are real and positive for this problem, the coefficients c, d may be chosen so that the ellipse is stretched along the real axis. For example, let us choose $d = 3.97$ for which $c = 3.95$. This should produce a stability region enclosing a highly elongated ellipse centered at $(3.97, 0)$, extending to $(7.93, 0)$. The resulting stability region is indicated in Fig. 3, and we see the expected shape is obtained. Since the domain is significantly extended in the x-direction, these coefficients should give excellent convergence results for systems whose eigenvalues are purely real. Indeed, since the domain is nearly four times longer along the x-axis than that for the second-order scheme in Fig. 1, we can increase the timestep accordingly and would expect a near fourfold acceleration in convergence to the steady state for such problems. Note that in order to produce stability regions that extend in the imaginary direction, imaginary values of c can be used, as was seen in Fig. 2 for $0 < d < 2$. If there are pure imaginary eigenvalues, then part of the imaginary axis should be contained in the stability region. This can be accomplished by selecting parameters γ_1, γ_2 based on the Van der Houwen polynomials rather than the Chebyshev polynomials [12], [16]. Finally, Faber polynomials can be used to enclose domains of arbitrary shape [15]. Next we demonstrate the effectiveness of the stability region shown in Fig. 3 for a representative diffusion problem.

8. Numerical example: Diffusion to steady state. As a numerical example, we use the two-stage RK scheme to solve the model diffusion problem

$$\frac{\partial u}{\partial t} = -\Delta u \quad \text{on} \quad \Omega = [0, 1] \times [0, 1] \tag{27}$$

with $u = 0$ on sides $x = 0, 1$, $y = 0$ and $u = 100$ on side $y = 1$. The RK scheme used is given by (9) with $s = 2$, $b_1 = 0$, $b_2 = 1$, and $c_2 = a_{21} = \gamma_2$ where the values of b_i have been chosen to enhance vector computations [9]. For spatial discretization we use second-order, central differences over a regular, uniform 41×41 grid. The initial iterate corresponds to $u = 0$ in the interior. The timestep is determined using a linear von Neumann stability analysis, which gives $\Delta t_{\max} = .5d/(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2})$, where $\Delta x = \Delta y = .025$ for this problem and $(d, 0)$ is the coordinate of the center of the associated ellipse.

The convergence histories for the residual iterates are compared in Fig. 4 for the second-order accurate scheme and the first-order accurate scheme with $d = 3.97$ ($\gamma_2 = .1260$) in

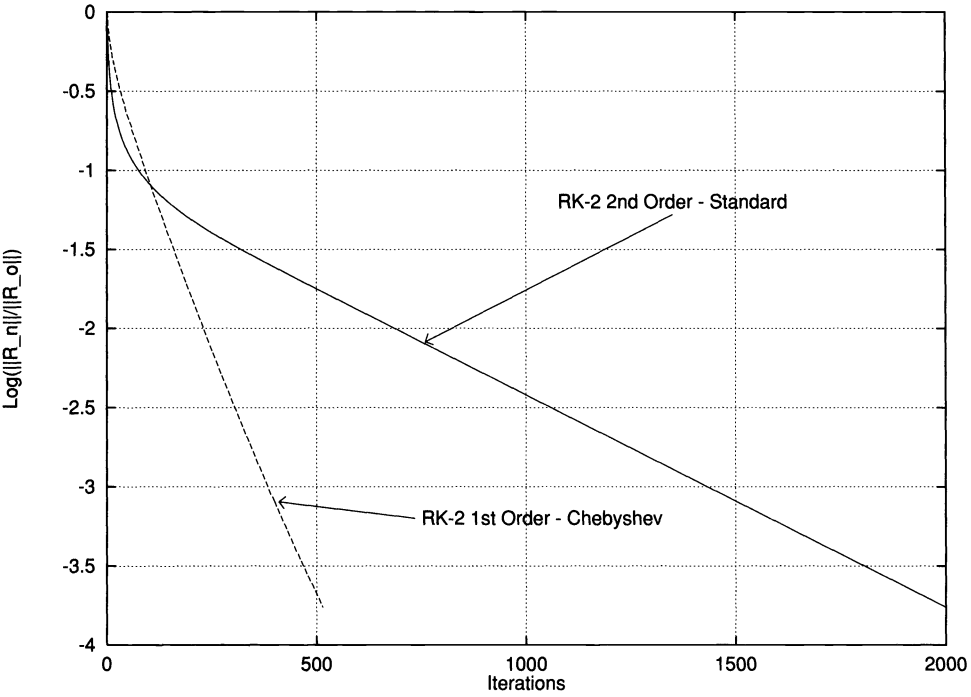


FIG. 4. *Convergence history for a diffusion equation. Standard and modified two-stage RK* 41 × 41 *grid.*

the Chebyshev-based method. It is seen that the Chebyshev-based parameters provide the expected fourfold improvement in the convergence rate.

9. Concluding remarks. As the above diffusion example indicates, the use of parameters chosen from iterative theory for Chebyshev-type methods produces a dramatic improvement in the convergence behavior for a representative two-stage RK scheme. These ideas are currently being extended and applied to nonlinear viscous flow applications and preliminary results have been excellent [9]. We remark, however, that some care should be exercised with regard to the nonlinear problem when multiple steady-state solutions arise. Clearly, choice of a scheme that is not time-accurate does correspond simply to an iterative recursion and there is no guarantee that a specific desired steady state will be obtained rather than another valid solution state. There are now numerous instances in engineering analysis where timestepping to a steady state is being advocated over other solution strategies. The present approach exploits parameter selection based on the equivalence to iterative recursion and the choice of best iteration parameters. This may accelerate the convergence of the recursion significantly as seen here. It is easy to "retrofit" the timestepping code to this form. Moreover, simply by use of a logical flag the algorithms may be selected for time-accurate analysis or faster convergence to steady state. In some situations it may even be desirable to use a hybrid scheme in which the early transient behavior is first monitored using the parameters corresponding to the high-order time-accurate scheme and then the parameters adaptively changed to correspond to the time-inaccurate recursion for fast steady-state solution. Of course, it is also tacitly assumed that a steady-state solution exists. Problems with periodic unsteady solutions must be accommodated using time-accurate integration. Finally, we remark that the approach here can be applied by introducing artificial transient terms for equilibrium equations and in conjunction with nonlinear continuation techniques. Conversely, in some instances it may be possible to modify an iterative solver and code to generate time-accurate transient solutions.

Acknowledgments. We would like to express our appreciation to the reviewers for their helpful suggestions.

REFERENCES

[1] S. Bova and G. F. Carey, *Local Time-Stepping Recursion with Adaptive Refinement for Steady-State Solutions of the Euler Equations*, 1994, manuscript.

[2] D. Chazan and W. Miranker, *Chaotic relaxation*, Linear Algebra Appl., 2 (1969), pp. 199–222.

[3] A. T. Chronopoulos and C. T. Pedro, *Iterative methods for nonsymmetric systems in DAEs and stiff ODEs codes*, Math. Comput. Simulation, 35 (1993), pp. 211–232.

[4] L. W. Ehrlich, *An ad hoc SOR method*, J. Comput. Phys., 44 (1981), pp. 31–44.

[5] R. A. Friesner, L. S. Tuckerman, B. C. Dornblaster, and T. V. Russo, *A method for exponential propagation of large systems of stiff nonlinear differential equations*, J. Sci. Comput., 4 (1989), pp. 327–354.

[6] E. Gallopoulis and Y. Saad, *Efficient solution of parabolic equations by polynomial approximation methods*, CSRD Report 969, Center for Supercomputing Research and Development, University of Illinois at Urbana-Champaign, February 1990.

[7] D. C. Jesperson, *A time-accurate multiple-grid algorithm*, AIAA Paper 85-1493-CP, in Proc. AIAA 7th Computational Fluid Dynamics Conference, Cincinnati, OH, July 1985, pp. 15–17.

[8] W. D. Joubert and T. A. Manteuffel, *Iterative methods for nonsymmetric linear systems*, in Iterative Methods for Large Linear Systems, D. R. Kincaid and L. J. Hayes, eds., Academic Press, Boston, 1990, pp. 149–171.

[9] A. A. Lorber and G. F. Carey, *A vector-parallel scheme for Navier–Stokes computations at multi-gigaflop performance rates*, International Journal for Numerical Methods in Fluids, 21 (1995), pp. 445–466.

[10] T. A. Manteuffel, *The Tchebychev iteration for nonsymmetric linear systems*, Numer. Math., 28 (1977), pp. 307–327.

[11] A. Pardhanani, *Grid Optimization and Multigrid Techniques for Fluid Flow and Transport Problems*, Ph.D. thesis, The University of Texas at Austin, December 1992.

[12] J. Pike and P. L. Roe, *Accelerated convergence of Jameson's finite-volume Euler scheme using Van Der Houwen integrators*, Comput. & Fluids, 13 (1985), pp. 223–236.

[13] A. Ralston and P. Rabinowitz, *A First Course in Numerical Analysis*, Second ed., McGraw-Hill, New York, 1987.

[14] R. E. Showalter, *Hilbert Space Methods for Partial Differential Equations*, Pitman, San Francisco, 1977.

[15] G. Starke and R. S. Varga, *A hybrid Arnoldi-Faber iterative method for nonsymmetric systems of linear equations*, Numer. Math., 64 (1993), pp. 213–240.

[16] P. J. Van der Houwen, *Construction of Integration Formulas for Initial Value Problems*, North-Holland, Amsterdam, 1977.

[17] R. S. Varga, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1962.

[18] D. M. Young, *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971.

[19] D. M. Young and R. T. Gregory, *A Survey of Numerical Mathematics*, Vols. 1 and 2, Dover, New York, 1972, 1973.

[20] D. M. Young and L. A. Hageman, *Applied Iterative Methods*, Academic Press, New York, 1981.

SIAM J. SCI. COMPUT.
Vol. 17, No. 1, pp. 78–89, January 1996

006

SOLUTION OF DENSE SYSTEMS OF LINEAR EQUATIONS IN THE DISCRETE-DIPOLE APPROXIMATION*

JUSSI RAHOLA†

Abstract. The discrete-dipole approximation (DDA) is a method for calculating the scattering of light by an irregular particle. The DDA has been used, for example, in calculations of optical properties of cosmic dust. In this method the particle is approximated by interacting electromagnetic dipoles. Computationally the DDA method includes the solution of large dense systems of linear equations where the coefficient matrix is complex symmetric. In this work, the linear systems of equations are solved by various iterative methods. QMR was found to be the best iterative method in this application. It converged in only a few more iterations than the full generalized minimal residual (GMRES) method. When the discretization of the particle was refined, the number of iterations remained constant even without preconditioning. The matrix–vector product in the iterative methods can be computed with the fast Fourier transform or the fast multipole algorithm. These algorithms make it feasible to solve dense linear systems of hundreds of thousands of unknowns.

Key words. electromagnetic scattering, system of linear equations, iterative methods, fast multipole method

AMS subject classifications. 65F10, 78A45

1. Introduction.

Light scattering calculations are widely applied in the astrophysics community. In many applications, it is necessary to be able to compute how an irregular dust grain scatters light. The scattering of light by individual dust grains seems to explain the optical properties of atmosphereless solar system bodies, such as the moon. This application of scattering calculations is described, for example, in Lumme and Rahola [20].

Purcell and Pennypacker [22] developed a method to compute the scattering of light by a particle of any shape. This method was later refined by Draine [7], who also called it the discrete-dipole approximation (DDA). The DDA is a rather crude way to discretize the integral equations that govern scattering. However, the DDA seems to work well in practice. The considerations for solving linear systems given here are not limited to the DDA, but apply also to other formulations of scattering calculations.

The DDA leads to a system of linear equations with a dense complex symmetric coefficient matrix. In previous studies, this system was usually solved by the conjugate gradient method applied to the normal equations. Lumme and Rahola [20] applied the quasi-minimal residual (QMR) method [13] to this problem and obtained a gain in performance.

The most time-consuming part in any iterative method for dense linear systems is the matrix–vector product. In the DDA algorithm, Goodman, Draine, and Flatau [16] showed how the fast Fourier transform (FFT) can be used in computing the matrix–vector product in some regular cubical geometries. Another fast method for computing the matrix–vector product is the fast multipole method [23], [17], [4], which we will apply to the DDA problem.

In §2, we will give the equations governing scattering and will review some of the properties of the coefficient matrix. In §3, several iterative methods will be tested. Section 4 describes various methods of computing the matrix–vector product. Section 5 shows how the fast multipole algorithm can be applied to the DDA problem.

2. The scattering equations.

In the DDA, the scatterer is approximated by interacting electric dipoles in an incident electromagnetic plane wave field that propagates with wave vector $\mathbf{k}$ and wave number $k = |\mathbf{k}|$. A harmonic time dependence $\exp(-i\omega t)$ is assumed. The dipole moments $\mathbf{p}_i$ and thus the electric fields $\mathbf{E}_i$ radiated by the dipoles are unknown.

*Received by the editors May 20, 1994; accepted for publication (in revised form) December 15, 1994.

†Center for Scientific Computing, P.O. Box 405, FIN-02101 Espoo, Finland (Jussi.Rahola@csc.fi).

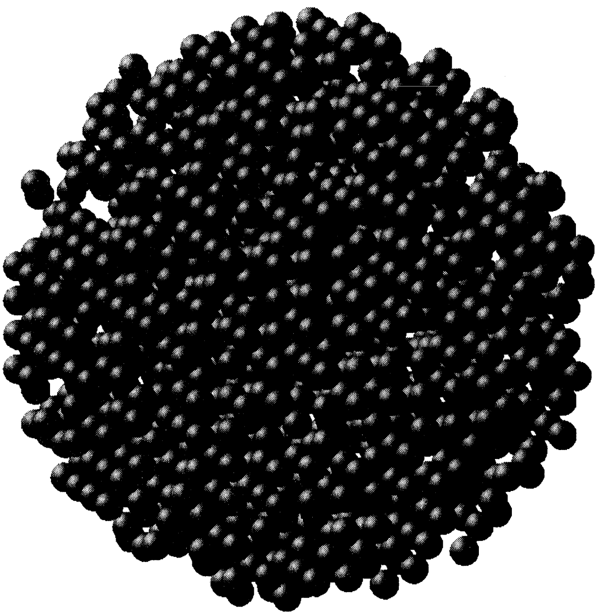


FIG. 1. *An example of a pseudosphere where a number of particles have been removed.*

At each dipole position $\mathbf{r}_i$, the electric field is determined by the incoming field $\mathbf{E}_i^0 = \mathbf{E}^0 \exp(i\mathbf{k} \cdot \mathbf{r}_i)$ plus the field radiated by the other dipoles. Thus the electric field at the dipoles can be solved from

$$\mathbf{E}_i = \mathbf{E}_i^0 + k^3\alpha \sum_{j \neq i}^{N_{\text{dip}}} \mathbf{T}_{ij} \cdot \mathbf{E}_j, i = 1, \ldots, N_{\text{dip}}, \tag{1}$$

where α is the polarizability of a single dipole. Each dipole is thought to represent a spherical volume of radius r_0. The polarizability of the dipole is connected to the index of refraction of the particle m and the dimensionless size parameter $x_0 = kr_0$ as given in [20]. The dipole interaction tensor (or dyadic) is given by

$$\begin{aligned} \mathbf{T}_{ij} &= \frac{e^{i\rho_{ij}}}{\rho_{ij}^3}(\rho_{ij}^2 + i\rho_{ij} - 1)\mathbf{1} + \frac{e^{i\rho_{ij}}}{\rho_{ij}^3}(-\rho_{ij}^2 - 3i\rho_{ij} + 3)\hat{\mathbf{r}}_{ij}\hat{\mathbf{r}}_{ij}, \\ \rho_{ij} &= k|\mathbf{r}_i - \mathbf{r}_j|, \end{aligned} \tag{2}$$

where $\mathbf{I}$ is the unit dyadic and $\hat{\mathbf{r}}_{ij}\hat{\mathbf{r}}_{ij}$ is the symmetric dyadic formed from the interdipole unit vectors $\hat{\mathbf{r}}_{ij} = (\mathbf{r}_i - \mathbf{r}_j)/|\mathbf{r}_i - \mathbf{r}_j|$.

The DDA is also called the coupled dipole method or the digitized Green function method. It can also be derived from a volume integral equation describing scattering. In [19], the DDA is compared with another computational technique, the method of moments, which better estimates the effect of the singularity of the free space dyadic Green function.

In our calculations, the two main geometries for the dipoles were a pseudospherical cluster of dipoles or a particle generated by the diffusion-limited aggregation process. The pseudospheres are generated by taking all the dipoles in a regular lattice that are within a given radius from the origin. An example of a pseudosphere from which a number of particles have been removed is given in Fig. 1. The diffusion-limited aggregation process builds fractal clusters by a three-dimensional random walk. A sample particle is given in Fig. 2.

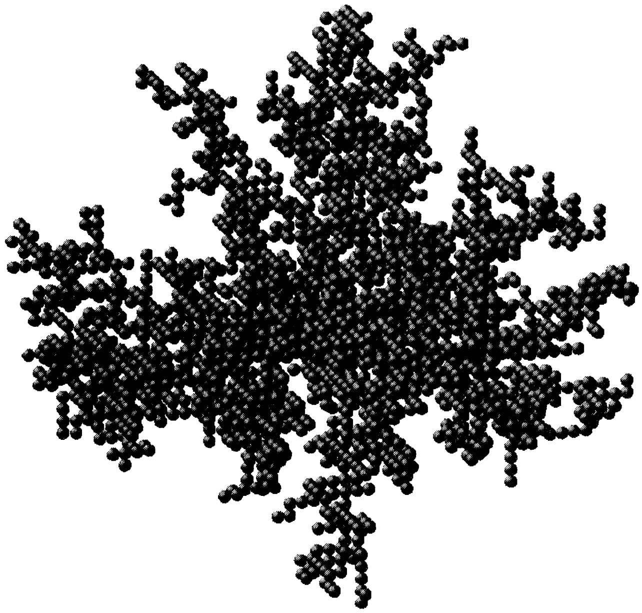


FIG. 2. *A particle generated by the diffusion-limited aggregation process.*

The equations (1) form a system of linear equations in the unknowns $\mathbf{E}_i$ where the coefficient matrix is complex symmetric. The right-hand side vector is given by the incident field $\mathbf{E}_i^0$. The coefficient matrix is full as each dipole interacts with all the other dipoles.

If we have N_{dip} dipoles, the system of linear equations has $3N_{\text{dip}}$ unknowns, because we need to determine the components of the electric field in the x, y, and z directions. When calculating the scattering of light by particles a few microns across, we typically need 1000–100,000 dipoles, giving rise to huge linear systems.

The spectrum of the matrix is shown in Fig. 3. Note that most of the eigenvalues lie on a line segment. This fact will be important in analyzing the convergence of iterative solvers.

3. Comparison of iterative solvers. A modern recipe for solving dense linear algebra systems is given by Edelman in his survey [8]: use a preconditioned iterative method that accesses the matrix only by matrix–vector multiplication and look for fast approximate methods for computing the matrix–vector product. Thus iterative methods can be used not only with sparse matrices but also with dense matrices. Actually, for very large dense problems, iterative methods are the only plausible solution methods. For an introduction to iterative methods, see [14].

In this work, several iterative methods were implemented and tested. The convergence of these methods depends on the geometry of the dipoles, the index of refraction m, and the size parameter x_0. The following methods are used: QMR, GMRES, conjugate gradient applied to the normal equations (CGNR), biconjugate gradient (BCG), conjugate gradient squared (CGS), and biconjugate gradient stabilized (BiCGStab). These methods belong to the family of Krylov-subspace methods, which access the matrix through matrix–vector multiplications. For a description of the methods, see [1]. We used the complex symmetric versions of QMR and BCG, which only need one matrix–vector multiplication per iteration [13]. Figure 4 shows the convergence of these methods in a typical small problem. The horizontal axis shows the number of matrix–vector products so it is roughly proportional to the CPU time. In the figure, the memory length of GMRES was 20.

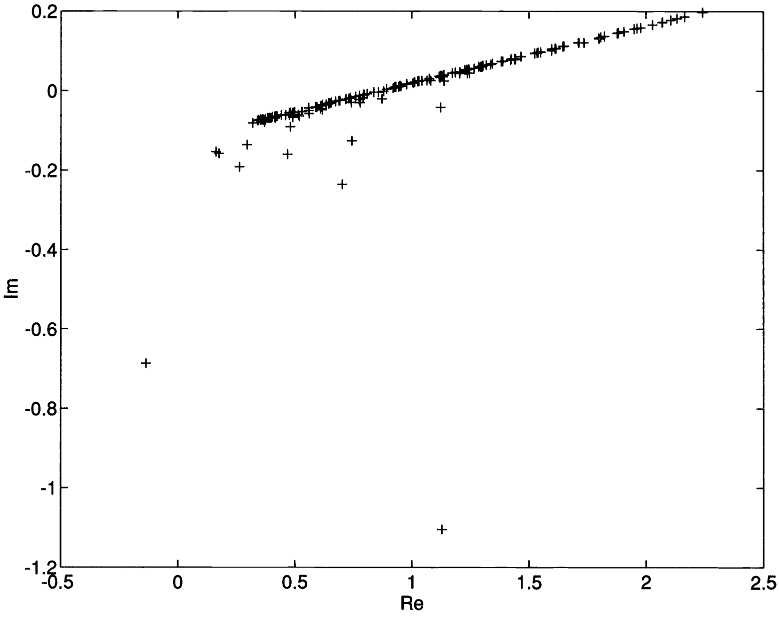


FIG. 3. *Spectrum of the linear system arising in the DDA calculations.*

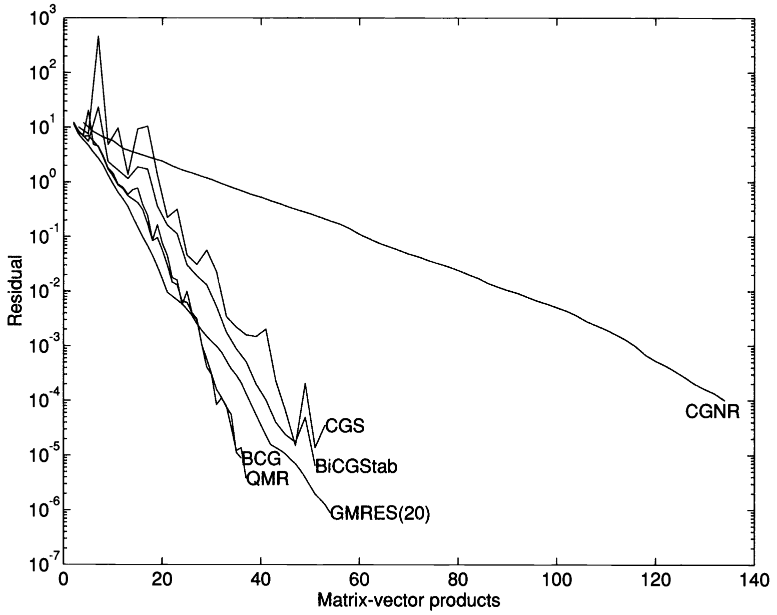


FIG. 4. *Convergence of various iterative methods.*

In bigger systems, CGS had trouble converging, and the residuals of BCG started to oscillate. CGNR was slower than the most efficient methods roughly by a factor of four. Figure 5 shows a comparison of QMR with GMRES using several memory lengths and with full GMRES. The different GMRES versions are converging with a linear rate after an initial sublinear phase. The full GMRES and QMR seem to find a superlinear rate of convergence,

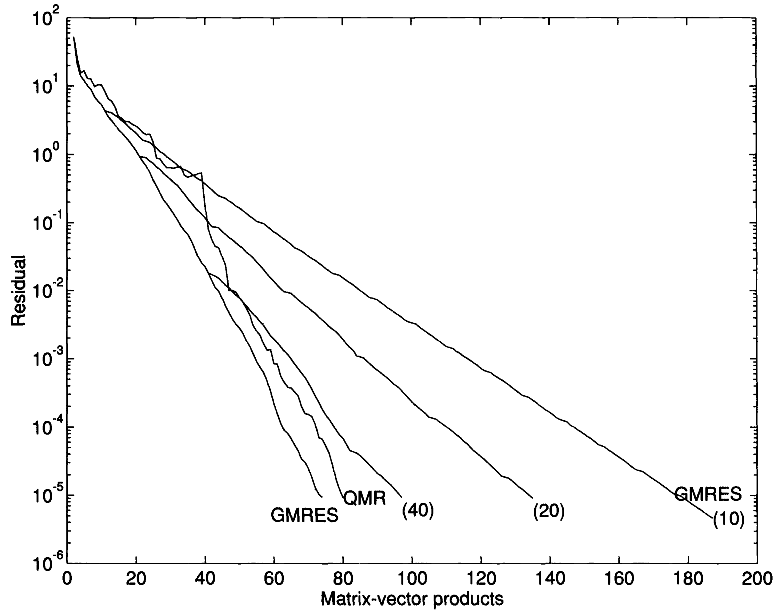


FIG. 5. *A comparison of QMR and GMRES with different memory lengths and with full GMRES.*

TABLE 1

The table shows the number of iterations for QMR (N_{iter}) and the elapsed CPU time in seconds (t) needed to reduce the initial residual by 10^{-5} when the same particle is discretized with different number of dipoles (N_{dip}) and with different size parameters (x_0) associated with each dipole. The number of unknowns is three times the number of dipoles N_{dip}.

N_{dip}	x_0	N_{iter}	t
32	0.3	6	0.23
136	0.188	7	0.46
304	0.144	7	1.52
1064	0.0947	7	3.3
2330	0.0731	7	10
5232	0.0555	7	21
10048	0.0448	7	37
20336	0.0355	7	91
137376	0.01875	7	562

but this may be just a better adaptation to the spectrum. The three iteration phases, sublinear, linear, and superlinear, are typically found when using Krylov-subspace methods [21].

QMR converges with only a few more iterations that full GMRES, which is optimal in the sense that it computes a minimal residual solution. However, full GMRES needs all the previously computed iterates and can thus be time and memory consuming. QMR is based on short recurrences and can be implemented efficiently. QMR was chosen to be the iterative method in the production version of this code.

Table 1 shows the number of QMR iterations needed to reduce the initial residual by a factor of 10^{-5} when the same particle is discretized with increasing resolution. This means that each dipole corresponds to a smaller volume, whose radius is given by the size parameter x_0. The geometry is a filled pseudosphere. The reported CPU time is on a Convex C3840 at the Center for Scientific Computing. The FFT was used to compute the matrix–vector product.

The number of iterations was remarkably constant. Note that in this simulation the index of refraction was smaller than in the earlier figures, resulting in faster convergence.

In the astrophysical calculations, the interest is in the scattering of light by micron-scale particles, leading to dipole configurations of tens or hundreds of thousands of dipoles. In practical calculations even smaller accuracies in the solution of the linear systems are acceptable because orientational averages of the results are taken.

An ideal iterative method for non-Hermitian matrices would have some minimization property and could be implemented using only a few of the latest iterates. Faber and Manteuffel have proved an important theorem for unsymmetric iterations [11], [12], which we state here in the form given in [14].

THEOREM 3.1. *Except for a few anomalies, ideal CG-like methods whose iterates are characterized by a minimal residual or an orthogonal residual property and that can be implemented based on short vector recursions exist only for matrices of the special form*

$$A = e^{i\theta}(T + \sigma I), \tag{3}$$

where T is Hermitian, $\theta \in \mathbb{R}$, $\sigma \in \mathbb{C}$.

In the DDA case, the spectrum of the coefficient matrix lies on a line except for a few eigenvalues, as visualized in Fig. 3. This structure of the spectrum probably accounts for the good performance of QMR. We plan to analyze the convergence of QMR and relate the spectrum of the coefficient matrix to the spectrum of the underlying continuous integral operator.

4. Calculating the matrix–vector product. The performance of iterative solvers for dense linear equations is determined by the number of iterations and the time it takes to multiply the coefficient matrix by a vector.

There are two evident ways to compute the matrix–vector product. The first one is simply to form the coefficient matrix and to keep it in memory. This is indeed the fastest way to calculate the matrix–vector product for small to medium-size matrices. For example, if half a gigabyte of central storage is available, dense matrices up to 7000 × 7000 elements can be stored using single precision. The second method, used for larger matrices, is to either store the matrix in the disk ("out of core") or recompute the matrix elements whenever necessary.

In order to be able to solve large dense linear systems one has to make use of the "all large dense matrices are structured" hypothesis [8], according to which nature does not just randomly throw n^2 numbers at us. Instead, there is some structure in most dense coefficient matrices and we must find an algorithm that exploits this structure.

In this paper, we have applied two fast methods, the FFT and the fast multipole method to compute the matrix–vector product.

Goodman, Draine, and Flatau [16] showed how to use the FFT in conjunction with the DDA. If the dipoles occupy positions in a regular cubical lattice and the lattice dimensions are doubled in each dimension, the matrix–vector multiplication reduces into a 3-D convolution. This convolution can be calculated using a 3-D FFT.

The use of FFT is not limited to cubical arrays of dipoles. The lattice spacings and extents can be different in different dimensions. Not all the lattice positions need to be occupied; the artificial dipoles affect only the performance of the Fourier transform. The size of the FFT is determined by first finding a lattice so that all the dipoles sit on that lattice. The smallest rectangular part of the lattice containing all the dipoles is taken as the computational lattice.

The use of the Fourier transform is very efficient if the dipoles are packed together in a dense and regular way. On the other hand, if the dipoles form a sparse dendrite-like particle generated, for example, by the diffusion-limited aggregation process, many artificial dipoles are needed in order to be able to use the FFT and the method slows down [20].

5. The fast multipole algorithm. The fast multipole method [23], [17], [4] is an efficient method to compute the electric potential of a system of N charges or multipoles. It can also be used in the solution of integral equations of potential theory [23], [24]. The fast multipole method was initially used in simulations where Coulomb or gravitational potential is used. In this paper we will apply the fast multipole method to compute the matrix–vector multiplication in electromagnetic scattering in three dimensions.

The brute force method of computing the interactions of N particles or the evaluation of a full matrix in integral equation computations requires $\mathcal{O}(N^2)$ operations. The full fast multipole method is a rather complicated algorithm that can achieve $\mathcal{O}(N)$ complexity in the solution of the Laplace equation or with the Coulomb or gravitational potential, and $\mathcal{O}(N \log N)$ complexity in the electromagnetic scattering case, i.e., in the solution of the Helmholz equation. For 2-D systems, the break-even point between the direct method and the fast multipole algorithm can be as low as 100 particles. In 3-D calculations, the break-even point is much larger, typically in the range 1000–10,000. An example of 3-D fast multipole computations using the Coulomb potential was given in [26].

For 2-D electromagnetic scattering problems, the fast multipole method was given in [9]. The 3-D version was given in [5], which formulates the scattering problem using a boundary integral equation. The DDA is related to a volume-integral equation. Thus we use different expansions of the potential than those found in [5].

5.1. The multipole and local expansions. Here we shall develop the multipole and local expansions for the dipole potential and show how the electric field and thus the matrix–vector product can be computed from these potentials. Here the matrix–vector product should be viewed as follows: given x, the current guess for the electric field caused by the dipoles, Ax gives the combined field of all the other dipoles at all the dipole positions. The electric field and the dipole moment of a dipole are related by $\mathbf{p}_i = k^3\alpha\mathbf{E}_i$.

The vector potential generated by an oscillating electric dipole at position $\mathbf{r}_i$ with dipole moment $\mathbf{p}_i$ is given by [18]

$$\phi(\mathbf{r}) = -ik\mathbf{p}_i \frac{e^{ik|\mathbf{r}-\mathbf{r}_i|}}{|\mathbf{r}-\mathbf{r}_i|}. \tag{4}$$

The electric field due to this potential is given by

$$\mathbf{E}(\mathbf{r}) = \frac{i}{k}\nabla \times (\nabla \times \phi(\mathbf{r})). \tag{5}$$

The expansion of the Green function occurring in the potential (4) in spherical coordinates is [18]

$$\frac{e^{ik|\mathbf{r}-\mathbf{r}_i|}}{|\mathbf{r}-\mathbf{r}_i|} = \begin{cases} 4\pi ik \displaystyle\sum_{n=0}^{\infty}\sum_{m=-n}^{n} h_n^{(1)}(kr)j_n(kr_i)Y_n^m(\theta,\phi)\left[Y_n^m(\theta_i,\phi_i)\right]^*, & r > r_i, \\ 4\pi ik \displaystyle\sum_{n=0}^{\infty}\sum_{m=-n}^{n} h_n^{(1)}(kr_i)j_n(kr)Y_n^m(\theta,\phi)\left[Y_n^m(\theta_i,\phi_i)\right]^*, & r < r_i. \end{cases} \tag{6}$$

The function $h_n^{(1)}$ is the spherical Hankel function of the first kind, j_n is the spherical Bessel function, and Y_n^m are the spherical harmonics.

In the fast multipole algorithm we need a multipole expansion, which can be used if we are evaluating the potential in a point that is further away from the origin than any of the dipoles. We also need a local expansion that is valid when all the dipoles are further away from the origin than the evaluation point.

The multipole coefficients of a collection of N_{dip} dipoles are given by

$$\text{(7)} \qquad \mathbf{M}_{nm} = \sum_{i=1}^{N_{\text{dip}}} \mathbf{p}_i j_n(kr_i) \left[Y_n^m(\theta_i, \phi_i)\right]^* .$$

The multipole expansion of the potential in terms of the multipole coefficients is

$$\text{(8)} \qquad \phi(\mathbf{r}) = 4\pi k^2 \sum_{nm} \mathbf{M}_{nm} h_n^{(1)}(kr) Y_n^m(\theta, \phi).$$

The local coefficients of a collection of N_{dip} dipoles are given by

$$\text{(9)} \qquad \mathbf{L}_{nm} = \sum_{i=1}^{N_{\text{dip}}} \mathbf{p}_i h_n^{(1)}(kr_i) \left[Y_n^m(\theta_i, \phi_i)\right]^* .$$

The local expansion of the potential is

$$\text{(10)} \qquad \psi(\mathbf{r}) = 4\pi k^2 \sum_{nm} \mathbf{L}_{nm} j_n(kr) Y_n^m(\theta, \phi).$$

When we are using the potential expansions in practice, the summations over n are truncated at a value N, while the summation over m is from $-n$ to n. For the expansion (8), N must be greater than kR, where $R = \max r_i$. For the expansion (10), N must be greater than kR_1, where R_1 is the maximum distance where $\psi(\mathbf{r})$ is going to be evaluated [25].

When given a potential (8) or (10) in terms of the multipole or local coefficients, we must be able to compute the electric field caused by this potential. For the multipole potential case, let us mark

$$\text{(11)} \qquad \gamma = h_n^{(1)}(kr) Y_n^m(\theta, \phi).$$

Now the electric field due to the multipole potential is given by

$$\text{(12)} \qquad \mathbf{E}(\mathbf{r}) = 4\pi i k \sum_{nm} \left\{ \mathbf{M}_{nm} \cdot \nabla\nabla\gamma + k^2 \gamma \mathbf{M}_{nm} \right\},$$

where $\nabla\nabla\gamma$ is a symmetric second-rank tensor whose representation is given in spherical coordinates by

$$\text{(13)} \qquad \begin{pmatrix} \frac{\partial^2 \gamma}{\partial r^2} & \frac{\partial}{\partial r}\left(\frac{1}{r}\frac{\partial \gamma}{\partial \theta}\right) & \frac{1}{\sin\theta}\frac{\partial}{\partial r}\left(\frac{1}{r}\frac{\partial \gamma}{\partial \phi}\right) \\ \frac{\partial}{\partial r}\left(\frac{1}{r}\frac{\partial \gamma}{\partial \theta}\right) & \frac{1}{r^2}\frac{\partial^2 \gamma}{\partial \theta^2} + \frac{1}{r}\frac{\partial \gamma}{\partial r} & \frac{1}{r^2}\frac{\partial}{\partial \theta}\left(\frac{1}{\sin\theta}\frac{\partial \gamma}{\partial \phi}\right) \\ \frac{1}{\sin\theta}\frac{\partial}{\partial r}\left(\frac{1}{r}\frac{\partial \gamma}{\partial \phi}\right) & \frac{1}{r^2}\frac{\partial}{\partial \theta}\left(\frac{1}{\sin\theta}\frac{\partial \gamma}{\partial \phi}\right) & \frac{1}{r^2 \sin^2\theta}\frac{\partial^2 \gamma}{\partial \phi^2} + \frac{1}{r}\frac{\partial \gamma}{\partial r} + \frac{\cos\theta}{r^2 \sin\theta}\frac{\partial \gamma}{\partial \theta} \end{pmatrix}.$$

The electric field due to the local potential is computed similarly.

5.2. Translations. In the fast multipole algorithm we need to translate the origin of a multipole or a local expansion and also to switch between multipole and local expansions. The translation formulas are based on the addition theorems for spherical wave functions [15], [27], [6]. They are stated here in a form that uses the Clebsch–Gordan or Wigner coefficients $C_{m_1 m_2 m}^{j_1 j_2 j}$ [2].

In the translation formulas, we use coefficients $c_{n\nu}^{m\mu p}$ defined by

$$c_{n\nu}^{m\mu p} = (-1)^m i^{n-\nu+p} \sqrt{\frac{4\pi(2n+1)(2\nu+1)}{2p+1}} C_{000}^{\nu n p} C_{\mu,-m,\mu-m}^{\nu n p}. \tag{14}$$

The first transformation is used to translate the origin of a multipole expansion. The new origin is given in spherical coordinates by (r_0, θ_0, ϕ_0),

$$\mathbf{M}'_{nm} = \sum_{\nu=0}^{\infty} \sum_{\mu=-\nu}^{\nu} \mathbf{M}_{\nu\mu} \sum_p c_{n\nu}^{m\mu p} j_p(kr_0) Y_p^{\mu-m}(\theta_0, \phi_0). \tag{15}$$

The second transformation is used to switch between a multipole and a local expansion

$$\mathbf{L}'_{nm} = \sum_{\nu=0}^{\infty} \sum_{\mu=-\nu}^{\nu} \mathbf{M}_{\nu\mu} \sum_p c_{n\nu}^{m\mu p} h_p^{(1)}(kr_0) Y_p^{\mu-m}(\theta_0, \phi_0). \tag{16}$$

Finally, this transformation is used to translate the origin of a local expansion

$$\mathbf{L}'_{nm} = \sum_{\nu=0}^{\infty} \sum_{\mu=-\nu}^{\nu} \mathbf{L}_{\nu\mu} \sum_p c_{n\nu}^{m\mu p} j_p(kr_0) Y_p^{\mu-m}(\theta_0, \phi_0). \tag{17}$$

In the above formulas, the summation over p takes the values $n+\nu$, $n+\nu-2, \ldots, \max(|m-\mu|, |n-\nu|)$. The translations are exact, but in practice they are truncated in a similar way as the potential expansions.

The above transformations are quite expensive to compute, especially when the order of the multipole expansion is high. Rokhlin [25] and Epton and Dembart [10] have developed a theory for the diagonal forms of the translation operators. The principal tool is the far-field signature function defined by

$$\hat{\phi}_{\mathbf{a}}(\mathbf{s}) = \lim_{r\to\infty} \left[\phi(\mathbf{a} + r\mathbf{s}) e^{-ikr} kr\right], \tag{18}$$

where $\mathbf{a}$ is the center of the expansion and $\mathbf{s}$ is a unit vector. This function can be directly constructed for the field of a collection of dipoles. On the other hand, given an expansion of the form (8), the truncated far-field signature function is given by

$$\hat{\phi}_{\mathbf{a},N}(\mathbf{s}) = 4\pi k^2 \sum_{n=0}^{N} \sum_{m=-n}^{n} \mathbf{M}_{nm} Y_n^m(\theta, \phi)/i^{n+1}, \tag{19}$$

where θ and ϕ are the spherical coordinates of $\mathbf{s}$.

The translation formula for the far-field signature function is [10]

$$\hat{\phi}_{\mathbf{b}}(\mathbf{s}) = e^{ik(\mathbf{b}-\mathbf{a})\,\mathbf{s}} \hat{\phi}_{\mathbf{a}}(\mathbf{s}), \tag{20}$$

where $\mathbf{b}$ is the new origin.

A truncated function describing the far-field behavior of the local expansion $\psi(\mathbf{r})$ is defined by

$$\hat{\psi}_{\mathbf{a},N'}(\mathbf{s}) = 4\pi k^2 \sum_{n=0}^{N'} \sum_{m=-n}^{n} \mathbf{L}_{nm} Y_n^m(\theta, \phi)/i^{n+1}. \tag{21}$$

The far-field signature function can be transformed into $\hat{\psi}_{\mathbf{c},N'}(\mathbf{s})$ by

$$\hat{\psi}_{\mathbf{c},N'}(\mathbf{s}) = M_L(\mathbf{s}, \mathbf{c} - \mathbf{b})\hat{\phi}_{\mathbf{b},N}(\mathbf{s}), \tag{22}$$

where

$$M_L(\mathbf{s}, \mathbf{r}) = \sum_{n=0}^{L}(2n + 1)i^n h_n^{(1)}(kr)P_n(\mathbf{s} \cdot \hat{\mathbf{r}}). \tag{23}$$

Here $L \geq N + N'$, P_n is the Legendre polynomial, and $\hat{\mathbf{r}}$ is the unit vector in the direction $\mathbf{r}$.

The function $\hat{\psi}_{\mathbf{c}}(\mathbf{s})$ obeys the translation theorem

$$\lim_{L \to \infty} \hat{\psi}_{\mathbf{d},L}(\mathbf{s}) = e^{ik(\mathbf{d}-\mathbf{c})\cdot\mathbf{s}}\hat{\psi}_{\mathbf{c},N}(\mathbf{s}). \tag{24}$$

In practice, the functions $\hat{\phi}_{\mathbf{a}}(\mathbf{s})$ and $\hat{\psi}_{\mathbf{b}}(\mathbf{s})$ are represented by a discrete set of K^2 points on the unit ball, where $K \approx kR_{\max}$ and $R_{\max}$ is the largest computational box in the fast multipole algorithm whose multipole and local expansions are used [25].

5.3. The fast multipole algorithm. The fast multipole algorithm describes an economical way of computing the electromagnetic interactions between the dipoles. The interactions between nearby dipoles are computed directly; the distant interactions are computed using truncated multipole and local expansions. In the algorithm, the computational domain is adaptively divided into boxes. The interaction between dipoles in two distant boxes is computed by forming the multipole expansions of the dipoles in the boxes, transforming the expansions into local expansions with respect to the origin of the other box, and evaluating the local expansions at the dipole positions.

The fast multipole algorithm organizes the evaluation of the dipole interactions so that the computations are made using the coarsest refinement level possible and maintaining a rigorous error bound. The number of terms used in the multipole calculations is determined by the size of the particle and is practically independent of the precision of the computations [25].

We will review the fast multipole algorithm as given in [4]. We shall be working with computational boxes b containing some number of dipoles. Each box is either a *parent box* if it is further divided in the algorithm or a *childless box*. *Colleagues* are adjacent boxes at the same refinement level. Each box has at most 26 colleagues.

The algorithm is given here in terms of the translation formulas (15)–(17). In an actual implementation, the diagonal forms of the translation theorems should be used.

ALGORITHM 5.1 (the fast multipole algorithm).

1. Refine the computational domain adaptively into boxes. If a box contains more than s dipoles, it is subdivided into eight child boxes.
2. Form the multipole coefficients (7) for each childless box. For each parent box, use (15) to translate the origin of the multipole expansions of each child to the center of the parent box and add the expansions to get the multipole expansion of the parent box.
3. For each childless box b, compute directly the interactions between the dipoles in b with the other dipoles in b and with the dipoles in the childless boxes that are adjacent to b.
4. For each box b, convert the multipole expansions of those children of colleagues of b's parent that are well separated from b into a local expansion about b's center using (16) and adding them up.
5. For each childless box b, using (12) at each dipole position calculate the electric field due to the multipole expansion of all descendants of b's colleagues whose parents are adjacent to b but who are not adjacent to b themselves.

6. If the multipole expansion of box b' was used at box b during stage 5, form the local expansion coefficients (9) of b at the center of b'.

7. For each parent box b, shift the center of the local expansion obtained in stages 4 and 6 to the center of b's children using (17).

8. For each childless box, calculate at all dipole positions the electric field due to the local expansion from stage 7 and add the contributions from stages 3 and 5 to get the full electric field caused by all the other dipoles.

At this point, we have limited practical experiences with the fast multipole methods. The current version that does not use the diagonal forms of the translation formulas becomes faster than direct calculations when using on the order of 10,000 dipoles. The calculations can be optimized by tabulating the values of the special functions. A lot of computing time is spent calculating the transformations (15)–(17) or their corresponding diagonal forms. There are only a fixed number of possible translations. The multipole-to-multipole translations are always from the child boxes to their parent box; the local-to-local translations are from a parent box to its children. There are 189 possible multipole-to-local translations in each refinement level, as indicated in [26]. These can be precomputed and tabulated.

6. Conclusion. We have presented an efficient way to solve the dense systems of linear equations arising in the discrete-dipole approximation. The recipe consists of using an efficient iterative method and calculating the matrix–vector products using a fast algorithm. In this project we chose QMR as the iterative method and showed how to apply the FFT and fast multipole techniques in this setting.

The same procedure can be applied in other electromagnetic scattering calculations where the physics are described by a volume integral equation.

The FFT gives by far the best method when the computational geometry is dense so that not very many artificial dipoles need to be used in the FFT algorithm. We plan to optimize the fast multipole method and compare its performance with the direct method and the FFT algorithm in a forthcoming study.

As the iterative methods converged in relatively few iterations, no preconditioning was considered necessary. A preconditioner could be built by running some inner iteration and computing the matrix–vector multiplication with the fast multipole algorithm using fewer terms in the multipole and local expansions.

In the scattering calculations, the same system of linear equations is solved many, maybe over one hundred, times. The convergence of the iterative solver could be speeded up by using a block version of QMR, as described in [3].

Acknowledgment. I would like to thank Kari Lumme for an introduction to the astrophysical problem and for his comments in applying the methods described here. Olavi Nevanlinna has provided important comments about the convergence of the iterative solvers and Olli Serimaa has given invaluable help with the special functions and the potential expansions. I would also like to thank the anonymous referees for their constructive comments.

REFERENCES

[1] R. Barrett et al., *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1993.

[2] L. C. Biedenharn and J. D. Louck, *Angular Momentum in Quantum Physics*, Addison-Wesley, Reading, MA, 1981.

[3] W. E. Boyse and A. A. Seidl, *A block QMR methodfor computing multiple simultaneous solutions to complex symmetric systems*, SIAM J. Sci. Comput., 17 (1996), pp. 263–274.

[4] J. Carrier, L. Greengard, and V. Rokhlin, *A fast adaptive multipole algorithm for particle simulations*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 669–686.

[5] R. COIFMAN, V. ROKHLIN, AND S. WANDZURA, *The fast multipole method for the wave equation: A pedestrian prescription*, IEEE Trans. Antennas and Propagation, 35 (3) (1993), pp. 7–12.

[6] O. CRUZAN, *Translational addition theorems for spherical vector wave functions*, Quart. Appl. Math., 20 (1962), pp. 33–40.

[7] B. T. DRAINE, *The discrete-dipole approximation and its application to interstellar graphite grains*, Astrophys. J., 333 (1988), pp. 848–872.

[8] A. EDELMAN, *Large dense numerical linear algebra in* 1993: *The parallel computing influence*, Internat. J. Supercomputer Appl., 7 (1993), pp. 113–128.

[9] N. ENGHETA, W. D. MURPHY, V. ROKHLIN, AND M. S. VASSILIOU, *The fast multipole method* (*FMM*) *for electromagnetic scattering problems*, IEEE Trans. Antennas and Propagation, 40 (1992), pp. 634–641.

[10] M. A. EPTON AND B. DEMBART, *Multipole translation theory for the* 3-*D Laplace and Helmholtz equations*, SIAM J. Sci. Comput., 16 (1995), pp. 865–897.

[11] V. FABER AND T. MANTEUFFEL, *Necessary and sufficient conditions for the existence of a conjugate gradient method*, SIAM J. Numer. Anal., 21 (1984), pp. 352–362.

[12] ———, *Orthogonal error methods*, SIAM J. Numer. Anal., 24 (1987), pp. 170–187.

[13] R. W. FREUND, *Conjugate gradient-type methods for linear systems with complex symmetric coefficient matrices*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 425–448.

[14] R. W. FREUND, G. H. GOLUB, AND N. M. NACHTIGAL, *Iterative solution of linear systems*, in Acta Numerica 1992, Cambridge University Press, 1992, pp. 57–100.

[15] B. FRIEDMAN AND J. RUSSEK, *Addition theorems for spherical waves*, Quart. Appl. Math., 12 (1954), pp. 13–23.

[16] J. J. GOODMAN, B. T. DRAINE, AND P. J. FLATAU, *Application of fast-Fourier-transform techniques to the discrete-dipole approximation*, Optics Letters, 16 (1991), pp. 1198–1200.

[17] L. GREENGARD AND V. ROKHLIN, *A fast algorithm for particle simulations*, J. Comput. Phys., 73 (1987), pp. 325–348.

[18] J. D. JACKSON, *Classical Electrodynamics*, 2nd ed., Wiley, New York, 1995.

[19] A. LAKHTAKIA AND G. W. MULHOLLAND, *On two numerical techniques for light scattering by dielectric agglomerated structures*, J. Res. Natl. Inst. Stand. Technol., 98 (1993), pp. 699–716.

[20] K. LUMME AND J. RAHOLA, *Light scattering by porous dust particles in the discrete-dipole approximation*, Astrophys. J., 425 (1994), pp. 653–667.

[21] O. NEVANLINNA, *Convergence of Iterations for Linear Equations*, Birkhäuser, Basel, 1993.

[22] E. M. PURCELL AND C. R. PENNYPACKER, *Scattering and absorption of light by nonspherical dielectric grains*, Astrophys. J., 186 (1973), pp. 705–714.

[23] V. ROKHLIN, *Rapid solution of integral equations of classical potential theory*, J. Comput. Phys., 60 (1985), pp. 187–207.

[24] ———, *Rapid solution of integral equations of scattering theory in two dimensions*, J. Comput. Phys., 86 (1990), pp. 414–439.

[25] ———, *Diagonal forms of translation operators for the Helmholtz equation in three dimensions*, Applied and Computational Harmonic Analysis, 1 (1993), pp. 82–93.

[26] K. E. SCHMIDT AND M. A. LEE, *Implementing the fast multipole method in three dimensions*, J. Statist. Phys., 63 (1991), pp. 1223–1235.

[27] S. STEIN, *Addition theorems for spherical wave functions*, Quart. Appl. Math., 19 (1961), pp. 15–24.

SIAM J. SCI. COMPUT.
Vol. 17, No. 1, pp. 90–99, January 1996

EQUIVARIANT PRECONDITIONERS FOR BOUNDARY ELEMENT METHODS*

JOHANNES TAUSCH†

Abstract. In this paper we propose and discuss two preconditioners for boundary integral equations on domains that are nearly symmetric. The preconditioners under consideration are equivariant; that is, they commute with a group of permutation matrices. Numerical experiments demonstrate their efficiency for the GMRES method.

Key words. preconditioners, integral equations, boundary element methods

AMS subject classifications. 65F10, 65R20, 65N38

1. Introduction. In the past few years symmetry-exploiting methods have become a popular topic in numerical linear algebra. Of special interest are linear systems that come from discretizing an integral equation defined on a domain with geometrical symmetries. These problems inherit the structure of the underlying group of symmetry transformations when the discretization is done in an appropriate way. By making use of this structure, it is possible to significantly reduce the amount of work involved in solving these systems.

Of course, this method will fail when the symmetry is destroyed by perturbing the domain slightly. Figures 1 and 2 show what kind of domains we have in mind. However, we expect that the symmetric and the perturbed problems are somewhat close to each other and that it is possible to take advantage of this situation as well.

Here we present an approach to how this can be achieved. We propose two preconditioners for the iterative solution of the discretized equation that have the structure of the related symmetry.

Now let us outline this paper. After a brief overview of boundary element methods and iterative methods for the linear system associated with the integral equation in §§2 and 3, we discuss in §4 symmetry reduction methods. In the following we will describe the preconditioners used. In §5 a preconditioner is constructed by discretizing the integral equation on a nearby symmetric surface. Section 6 contains a preconditioner that is the solution of a minimization problem. Some results of numerical experiments are presented in §7.

2. Boundary integral methods. Consider the linear integral equation

$$\lambda\rho(x) + \mathcal{K}\rho(x) = g(x), \qquad x \in \mathcal{B}, \tag{1}$$

with the boundary integral operator

$$\mathcal{K}\rho(x) = \int_{\mathcal{B}} k(x, y)\rho(y)\, d\mathcal{B}(y) \tag{2}$$

defined on a linear space of functions on $\mathcal{B}$, which is denoted by X.

We are interested in the case when $\mathcal{B}$ is a closed and compact surface in the three-space $\mathcal{B} \subset \mathbb{R}^3$. Equations like (1) arise from solving Laplace's equation on a domain with boundary $\mathcal{B}$, in which case the kernel $k(x, y)$ is weakly singular.

One of the standard approaches for solving (1) numerically is the collocation method. The idea here is to seek an approximation of the solution ρ in an n-dimensional linear subspace $X_n \subset X$ spanned by basis functions $\{\phi_i\}_{i\in\mathcal{N}}$, where $\mathcal{N} = \{1, \ldots, n\}$.

*Received by the editors May 18, 1994; accepted for publication (in revised form) February 2, 1995. This work was supported by the National Science Foundation grant DMS-9403392.

†Department of Mathematics, Colorado State University, Fort Collins, CO 80523 (Tausch@Math.Colostate.Edu).

In general, the solution of the integral equation (1) will not be a member of X_n, and therefore a function in this subspace cannot solve the equation at all points on the surface. Instead, one forces the unknown function ρ_n to satisfy (1) at least at points $\{p_i\}_{i\in\mathcal{N}} \subset \mathcal{B}$, which are called collocation points.

If ρ_n is written as a linear combination of the basis functions $\rho_n = \sum c_i\phi_i$, then the collocation method yields a linear system for the coefficients $x = (c_1, \ldots, c_n)^T$ of the following form:

$$Ax = b, \tag{3}$$

where the entries of the matrix are given by

$$A(i, j) = \lambda\phi_j(p_i) + (\mathcal{K}\phi_j)(p_i) \tag{4}$$

and the right-hand side is given by

$$b(i) = g(p_i)\,.$$

If the collocation points are chosen in a way such that the interpolation problem

$$\text{find } \rho_n \in X_n \text{ such that } \rho(p_i) = \rho_n(p_i) \quad \text{for } i \in \mathcal{N}$$

is uniquely solvable for all functions $\rho \in X$, then the system (3) is nonsingular.

Typically, the subspace X_n consists of functions that are piecewise polynomial on a triangulation of $\mathcal{B}$; this has been extensively studied by Atkinson [3]. For estimates of the discretization error $|\rho_n - \rho|$ we refer to the vast literature on numerical methods for integral equations; see, e.g., [4] or [13].

3. GMRES. The matrix of the collocation method in (3) is, in general, dense and nonsymmetric. A classic approach for solving (3) is the multigrid method; see, for instance, [12] or [13]. This technique works well when the operator $\mathcal{K}$ in the integral equation (1) is compact. If the boundary is only piecewise smooth, then this assumption is often violated and the multigrid iteration may perform poorly or may even diverge [5].

Only recently other iterative methods, like conjugate gradients and Krylov suuspace methods, have been studied in connection with boundary integral equations; see, e.g., [8] and [19]. One of these methods is the GMRES method of Saad and Schultz [14].

A well-known technique to improve the convergence of iterative methods is preconditioning; see, e.g., [11]. The idea here is to multiply the linear system $Ax = b$, with the inverse of a nonsingular matrix L, the preconditioner, and to solve the equivalent linear system $L^{-1}Ax = L^{-1}b$. Alternatively, one can transform the unknown x to $Lx = y$ and solve $AL^{-1}y = b$. Note that the products $L^{-1}A$ or AL^{-1} are actually never formed—this would be too costly. Instead, in each step of the iteration an additional linear system with the preconditioner has to be solved. Hence it should be possible to do this efficiently. To get an overall saving, the preconditioner should also decrease the number of steps involved in solving the proposed problem.

A good choice for a preconditioner is a matrix that factors A in the form $L^{-1}A = I + B$ where the norm of the remainder $\|B\|_2$ is small. Then the eigenvalues of the preconditioned matrix $L^{-1}A$ cluster around unity and yield a small condition number.

Another suitable preconditioner is a matrix L that factors A to a low-rank perturbation of the identity; i.e., $L^{-1}A = I + R$. Here R denotes a low-rank matrix: $rk(R) = p \ll n$ and no assumption must be made about the norm of R. In this case the preconditioned GMRES iteration terminates after at most p steps with the exact solution.

4. Equivariance. We are interested in the structure of the system matrix in (3) when the surface $\mathcal{B}$, on which the integral equation (1) is to be solved, has symmetries. That is, $\mathcal{B}$ is left invariant under a group Γ of isometries such as rotations and reflections. Each isometry $\gamma \in \Gamma$ gives rise to a linear operator $\Pi_\gamma : X \to X$ mapping the function $f \in X$ on $f \circ \gamma^{-1} \in X$. The operators $\{\Pi_\gamma : \gamma \in \Gamma\}$ defined in this way form a group under the usual multiplication of operators, which is isomorphic to the group of isometries. Henceforth we will use the same symbol for both groups.

Moreover, suppose the kernel of the integral operator $\mathcal{K}$ in (2) depends only on the distance of the two points x and y; i.e.,

$$k(x, y) = \tilde{k}(|x - y|)\,.$$

This situation is typical when equation (1) comes from an integral reformulation of a boundary value problem. Using a change of variables, it is straightforward that such an integral operator commutes with the action of an element in Γ; i.e.,

$$\mathcal{K}\Pi_\gamma = \Pi_\gamma \mathcal{K}\,. \tag{5}$$

In general, an operator $\mathcal{K}$ with the property (5) is called *equivariant with respect to the group action* Γ, or simply Γ*-equivariant.*

In order to be able to exploit the structure of the continuous problem, the discretization must not destroy the symmetry. It is required that the basis functions and the collocation points remain invariant under the action of the group. This is the content of our assumption.

ASSUMPTION 4.1. *The group* Γ *defines a group of permutations on the indices*

$$\gamma\colon (1, \ldots, n) \mapsto (\gamma 1, \ldots, \gamma n),$$

where the action of the group element γ *on the index* i *is defined by*

$$\Pi_\gamma \phi_i = \phi_{\gamma i}$$

and

$$\gamma p_i = p_{\gamma i}.$$

Moreover, the group of permutations is isomorphic to Γ.

The permutations on the indices in turn induce permutation matrices P_γ in the usual way:

$$(P_\gamma b)(i) = b(\gamma^{-1} i)$$

for a vector $b \in \mathbb{R}^n$. Using the equivariance of $\mathcal{K}$ and the above-defined group actions we obtain for the matrix entries in (4):

$$A(\gamma^{-1}i, j) = (\mathcal{K}\phi_j)(p_{\gamma^{-1}i}) = (\Pi_\gamma \mathcal{K}\phi_j)(p_i) = (\mathcal{K}\Pi_\gamma \phi_j)(p_i) = (\mathcal{K}\phi_{\gamma j})(p_i) = A(i, \gamma j)\,.$$

Hence we see that the matrix A has the three equivalent properties:

$$A(\gamma^{-1}i, j) = A(i, \gamma j) \qquad \forall i, j \in \mathcal{N}, \gamma \in \Gamma, \tag{6}$$

$$A(i, j) = A(\gamma i, \gamma j) \qquad \forall i, j \in \mathcal{N}, \gamma \in \Gamma, \tag{7}$$

$$P_\gamma A = A P_\gamma, \qquad \gamma \in \Gamma\,. \tag{8}$$

Equation (8) is the discrete analogue to equation (5), and, consequently, a matrix with property (8) is called Γ-equivariant. For the following, let us denote an equivariant matrix by L.

The most prominent examples of this class are the circulant matrices, which are equivariant with respect to the cyclic group $\mathbf{Z}_n$. In this case the group action is given by $\gamma i = (\gamma + i) \bmod n$ for $\gamma \in \mathbf{Z}_n$ and $i \in \mathcal{N}$ [7]. It is well known that linear systems with circulant matrices can be solved efficiently by means of the discrete Fourier transform.

It is possible to generalize this idea to arbitrary equivariant matrices. The key here lies in the irreducible representations of the group Γ; for an introduction to representation theory we refer to [15]. The irreducible representations determine a sparse and unitary matrix F—the generalized Fourier matrix—which factors L in the form $L = F^H \widehat{L} F$, with $\widehat{L}$ being a block diagonal matrix. Since F is a unitary matrix, its inverse is given by the Hermitian transpose; i.e., $F^{-1} = F^H$. Instead of solving the linear system $Lx = b$, the equivalent system $\widehat{L}\widehat{x} = \widehat{b}$ with right-hand side $\widehat{b} = Fb$ and unknown $\widehat{x} = Fx$ is solved and then the solution x is recovered from $\widehat{x}$ via the inverse transformation $x = F^H\widehat{x}$.

The diagonal blocks of $\widehat{L}$ can be computed with n^2 floating point operations, the transformations $\widehat{b} = Fb$, and $x = F^H\widehat{x}$ can be done in $\#(\Gamma)n$ flops. Thus the overhead compares to one matrix–vector multiplication and is negligible, considering that L is a full matrix. The major savings in computational effort comes from the fact that a number of small systems are solved as opposed to one big system in the unreduced case. The size of these systems decreases with the order of the group.

For more details about this generalization of the Fourier transform we refer to [16]. An earlier description of symmetry reduction using projections can be found in [1] and [10]. The approach presented there yields equivalent subsystems. Applications to boundary element methods with numerical results are in [2] and in [20].

5. Preconditioner derived from a nearby symmetric surface. In the next two sections we describe two preconditioners to handle problems defined on domains that are close to a domain with geometrical symmetries.

The first preconditioner is obtained by discretizing the integral equation on the symmetric surface. As it was pointed out in the previous section, this yields an equivariant matrix, which can be inverted efficiently.

Let us briefly discuss how this preconditioner is constructed. Usually surfaces are represented via parameterizations. Since we want to include piecewise smooth surfaces, it is convenient to use a piecewise linear (PL) manifold for the domain of the parameterization. Here we recall the definition found in Georg [9]: A *PL-manifold* is a finite collection $\mathcal{S}_{PL}$ of closed triangles in $\mathbb{R}^3$, each having affinely independent vertices $T_i = [v_0^i, v_1^i, v_2^i]$, $i = 1, \dots, J$. In addition, we require the following.

1. The intersection of two triangles in $\mathcal{S}_{PL}$ is empty or a vertex or an edge.
2. Each edge is common to exactly two triangles.

We assume that it is possible to parameterize the symmetric as well as the perturbed surface with the same PL-manifold. Then our discretization scheme can be described as follows:

1. Find a symmetry-respecting PL-manifold $\mathcal{S}_{PL}$ and parameterizations (i.e., piecewise smooth isomorphisms) $m : \mathcal{S}_{PL} \to \mathcal{B}$ and $\tilde{m} : \mathcal{S}_{PL} \to \tilde{\mathcal{B}}$ of the symmetric and the perturbed surface, respectively.

2. Define a set of basis functions $\{\psi_1, \dots, \psi_n\}$ on $\mathcal{S}_{PL}$. Lifting them to the surfaces $\mathcal{B}$ and $\tilde{\mathcal{B}}$ via $\psi_i = \phi_i \circ m$ and $\psi_i = \tilde{\phi}_i \circ \tilde{m}$ produces basis functions $\{\phi_1, \dots, \phi_n\}$ and $\{\tilde{\phi}_1, \dots, \tilde{\phi}_n\}$. Note that the basis on $\mathcal{B}$ must respect the symmetry in the sense of our assumption.

3. Define collocation points $\{q_1, \dots, q_n\}$ on $\mathcal{S}_{PL}$ and map them to the two surfaces: $p_i = m(q_i)$ and $\tilde{p}_i = \tilde{m}(q_i)$.

Applying the collocation method, we obtain the two nonsingular matrices L and A arising from the symmetric and the unsymmetric problem, respectively. The matrix L is only a good preconditioner when the quantity $\|A - L\|$ is small. In [17] it is shown that this is in fact true when the parameterizations and their first few derivatives are close to each other.

The use of the above preconditioner has a significant drawback. Usually the assembly of the system matrix is the most costly part of a boundary element technique. Our method requires two matrices—one for the symmetric and one for the perturbed problem. Even though only a fraction of matrix entries have to be determined in the equivariant case, the savings in the iteration may not justify the extra calculation of the preconditioner.

This objection, however, does not apply for situations where the perturbed surface differs from the symmetric surface only on a small piece. In this case the respective basis functions and the collocation points are identical except for a few, which implies that only a few rows and columns of A have to be updated to obtain the preconditioner. In other words, A is a low-rank perturbation of L or $L^{-1}A$ is a low-rank perturbation of the identity. This in turn implies that the number of steps in the GMRES iteration of $L^{-1}A$ is bounded by this rank.

In the next section we introduce a preconditioner that does not depend on additional surface integrations.

6. An optimal preconditioner. Our next goal is to find an equivariant matrix L so that the product $L^{-1}A$ is as close to the identity as possible. In other words, the preconditioner has to minimize the quantity $\left\|L^{-1}(A - L)\right\|$ in a consistent matrix norm. An upper bound for this number can be obtained easily. Setting $B = A - L$, we estimate

$$\left\|L^{-1}B\right\| = \left\|(A - B)^{-1}B\right\| = \left\|(I - A^{-1}B)^{-1}A^{-1}B\right\| \leq \frac{\left\|A^{-1}B\right\|}{1 - \left\|A^{-1}B\right\|}.$$

Since the right-hand side of this inequality is monotonically increasing with $\left\|A^{-1}B\right\|$, it is straightforward to minimize $\|B\| = \|A - L\|$. Thus we require that our preconditioner solves the following minimization problem:

$$L \equiv \min\{\|A - L\| : L \text{ is } \Gamma\text{-equivariant}\}\ . \tag{9}$$

We will show that this problem is trivially solvable in the Frobenius norm: to obtain the optimal preconditioner, one has to average over the orbits of the group action on the indices. This construction can be viewed as a generalization of T. Chan's circulant preconditioner for Toeplitz systems [6]. Here this idea is extended to any finite group. More formally, we have Theorem 6.1.

THEOREM 6.1. *The matrix defined by*

$$L(i, j) = \frac{1}{\#\Gamma}\sum_{\gamma\in\Gamma} A(\gamma i, \gamma j) \tag{10}$$

is Γ*-equivariant and is the optimal solution of the minimization problem* (9) *in the Frobenius norm. In matrix notation, the optimal preconditioner has the form*

$$L = \frac{1}{\#\Gamma}\sum_{\gamma\in\Gamma} P_{\gamma^{-1}} A P_{\gamma}. \tag{11}$$

Before we give the proof of the above theorem, we need some simple notions of a group acting on a set of integers.

Consider the index i in the index set $\mathcal{N}$. We denote the set $\mathrm{Orb}(i) := \{\gamma i : \gamma \in \Gamma\}$ the *orbit* of the group action on the index i. The set $\mathcal{N}$ can be viewed as the disjoint union of its orbits. A subset $\mathcal{S} \subset \mathcal{N}$ that contains exactly one element from each orbit is called a *selection* of indices. The *isotropy subgroup* of the index i consists of the group elements that leave i fixed; i.e.,

$$\mathrm{Fix}(i) = \{\gamma \in \Gamma : \gamma\, i = i\}\,.$$

If the isotropy subgroup of an index is trivial, i.e., $\mathrm{Fix}(i) = \{e\}$, then the size of the orbit containing i is equal to the group order. If there are fixed points in the group action then the size of the orbit is given by $\#\mathrm{Orb}(i) = \#\Gamma / \#\mathrm{Fix}(i)$.

Now we are able to prove the above theorem.

Proof. Let L be an arbitrary Γ-equivariant matrix. The main idea of the proof is to rearrange the summation in the Frobenius norm

$$\|A - L\|_F^2 = \sum_{i' \in \mathcal{N}} \sum_{j' \in \mathcal{N}} \left(A(i', j') - L(i', j')\right)^2$$

so that the orbits of $L(i, j)$ stand together. We write the index j' as the product of a group element and an index from a selection, i.e., $j' = \gamma j$ for a $\gamma \in \Gamma$ and $j \in \mathcal{S}$. When summing up over all $\gamma \in \Gamma$ and $j \in \mathcal{S}$ there might be repetitions due to fixed points in the group action. To account for that we have to divide by the order of the isotropy subgroup. Thus we obtain

$$\begin{aligned}\|A - L\|_F^2 &= \sum_{i' \in \mathcal{N}} \sum_{\gamma \in \Gamma} \sum_{j \in \mathcal{S}} \frac{1}{\#\mathrm{Fix}(j)} \left(A(i', \gamma j) - L(i', \gamma j)\right)^2 \\ &= \sum_{i \in \mathcal{N}} \sum_{\gamma \in \Gamma} \sum_{j \in \mathcal{S}} \frac{1}{\#\mathrm{Fix}(j)} \left(A(\gamma i, \gamma j) - L(\gamma i, \gamma j)\right)^2 .\end{aligned}$$

In the last step we have set $i = \gamma^{-1} i'$, which changes only the sequence of the summation. Finally, using the equivariance of L, we get

$$\|A - L\|_F^2 = \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{S}} \frac{1}{\#\mathrm{Fix}(j)} \sum_{\gamma \in \Gamma} \left(A(\gamma i, \gamma j) - L(i, j)\right)^2 .$$

The above expression consists of uncoupled optimization problems for the $L(i, j)$. The optimal solution can be obtained by minimizing each term

$$\sum_{\gamma \in \Gamma} \left(A(\gamma i, \gamma j) - L(i, j)\right)^2 .$$

This yields equation (10):

$$L(i, j) = \frac{1}{\#\Gamma} \sum_{\gamma \in \Gamma} A(\gamma i, \gamma j)\,.$$

It is easy to see that L is equivariant. Equation (11) can be verified by pre- and postmultiplying with canonical basis vectors. □

Computing one entry $L(i, j)$ by formula (10) takes $\#\Gamma$ additions. Since the matrix L is determined by the indices $i \in \mathcal{S}$ and $j \in \mathcal{N}$ this adds up to n^2 floating point operations for the whole matrix. Thus the calculation of the optimal preconditioner costs approximately about as much as one single matrix–vector multiplication. This is clearly an advantage over the preconditioner of the previous section, because no costly surface integrations have to be done.

TABLE 1
Iteration results for the second kind equation on the ellipsoid.

	No precond.		Opt. precond.	
N	*its*	δ	*its*	δ
48	11	.050	5	.0055
192	10	.042	5	.0095
768	11	.048	6	.0083

Note that the matrix constructed in the above theorem is not always nonsingular. However, if A is positive, i.e., $\langle Ax, x\rangle > 0$ for $x \neq 0$, then the preconditioner is positive as well, as it is shown in the following calculation:

$$\langle Lx, x\rangle = \frac{1}{\#\Gamma}\sum_{\gamma\in\Gamma}\langle P_{\gamma^{-1}}AP_\gamma x, x\rangle = \frac{1}{\#\Gamma}\sum_{\gamma\in\Gamma}\langle AP_\gamma x, P_\gamma x\rangle .$$

Here we have used that the permutations P_γ form a group of orthogonal matrices; thus $P_{\gamma^{-1}} = P_\gamma^{-1} = P_\gamma^T$. The last expression is positive when $x \neq 0$ by the positivity of A.

This generalizes an analogous result for circulant matrices [18].

7. Some numerical results.

7.1. Second kind equations. In the following we present some numerical results using the optimal preconditioner described above. The first examples deal with the integral equation

$$2\pi\rho(x) + \int_{\mathcal{B}} \frac{\partial}{\partial n_y}\left(\frac{1}{|x-y|}\right)\rho(y)d\mathcal{B}(y) + (2\pi - \Omega(x))\rho(x) = g(x),$$

which comes from solving Laplace's equation on a domain with boundary surface $\mathcal{B}$. Here $\Omega(x)$ denotes the solid angle of $\mathcal{B}$ at the point x, which is 2π when the surface is smooth at x. The linear systems come from collocating with functions that are piecewise constant on a triangulation of the parameter space.

The GMRES iteration was continued until the 2-norm of the residual was reduced by the factor 10^{-15}. The tables show the number of iterations (its) as well as the average reduction factor of the residual in each step (δ) for various refinements of the grid (N denotes the number of triangles). The parameters its and δ are compared for the original and the preconditioned system.

The domain in the first example is the ellipsoid $(x/1.1)^2 + (y/1.05)^2 + z^2 = 1$. The parameterizing $\mathcal{S}_{PL}$-manifold is the unit cube, which induces a group action of order 48 on the indices. This group action was used to construct the preconditioner as in our theorem (10). The results of the experiments are shown in Table 1. In the middle column are the results of the iteration applied on the original system; in the right column are the respective numbers for the preconditioned system.

The second domain is a cube, with one side perturbed by a quadratic surface; see Fig. 1. Due to the edges, the number of steps of the unpreconditioned iteration is higher than in the previous example with a smooth surface. Note, however, that the increase of steps in the preconditioned method is not so significant. The numerical results are displayed in Table 2.

In the third example (a cube with a small cube removed in one corner, cf. Fig. 2) the preconditioner performed poorly. This is due to the fact that the derivatives of the parameterizations are not nearby, yielding boundary integral operators that are not close in some norm. The results are shown in Table 3.

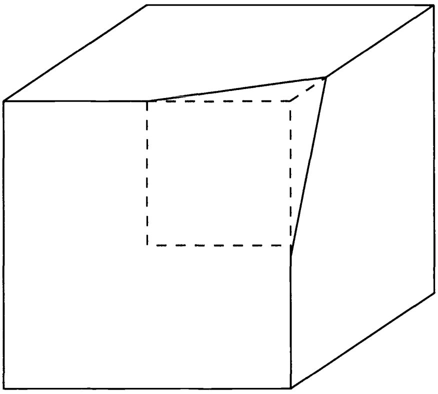


FIG. 1. *The perturbed cube.*

TABLE 2
Iteration results for the second kind equation on the perturbed cube.

	No precond.		Opt. precond.	
N	*its*	δ	*its*	δ
48	19	.176	6	.014
192	22	.219	7	.028
768	22	.235	7	.033

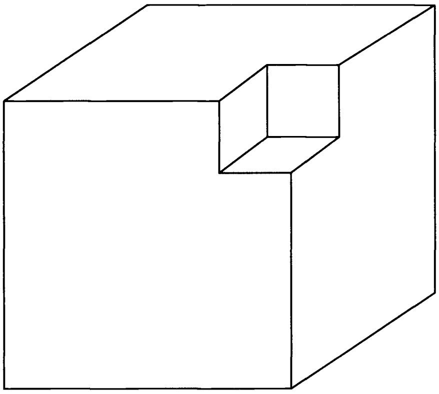


FIG. 2. *The cube with a small cube removed in one corner.*

TABLE 3
Iteration results for the second kind equation on the domain of Fig. 2.

	No precond.		Opt. precond.	
N	*its*	δ	*its*	δ
384	22	.24	16	.20
1152	23	.26	18	.23

The same experiments with a preconditioner that comes from a surface with symmetries did not reveal noticeable differences. The optimal preconditioner performed only slightly better.

TABLE 4
Iteration results for the first kind equation on the perturbed cube.

	No precond.		Opt. precond.	
N	*its*	δ	*its*	δ
48	23	.250	8	.039
192	33	.373	10	.068
768	44	.495	10	.084

TABLE 5
Iteration results for the first kind equation on the domain of Fig. 2.

	No precond.		Opt. precond.	
N	*its*	δ	*its*	δ
384	39	.450	9	.064
1152	49	.541	10	.087

7.2. First kind equations. In the above numerical examples the preconditioner reduced the number of steps in the iteration significantly; however, the unpreconditioned iteration converges reasonably fast as well, especially in the case of the ellipsoid. This is due to the well-posed nature of second kind equations (1) with compact operators.

This picture changes when first kind equations are to be solved. Equations of this type are of great importance for boundary element methods. We experimented with the single layer operator from potential theory, which is defined by

$$\mathcal{S}\rho(x) = \int_{\mathcal{B}} \frac{1}{|x-y|}\, \rho(y)\, d\mathcal{B}(y)\,.$$

Since this operator is compact, its inverse is not bounded and the equation $\mathcal{S}\rho = g$ is ill posed. When a discretization technique is applied then the number of GMRES iterations will increase with the refinement of the mesh.

The experiments suggest that our preconditioners work well especially for first kind equations. Compare with the results of Table 4, which were obtained by discretizing the perturbed cube of Fig. 1 in the same way as for the double layer equation.

As expected, the number of iterations increases as the grid is refined. This is also the case when the preconditioner is used; however, the increase is much slower. The two preconditioners that we have discussed perform almost equally well as in the case of the double layer equation.

Surprisingly, we obtained good performance for the single layer equation even on the domain of Fig. 2, where preconditioning of the double layer equation failed; see Table 5. This behavior may be attributed to the higher sensitivity of the double layer operator to perturbations of the surface. We will investigate this in our future work [17].

Acknowledgment. The author would like to thank his Ph.D. advisors Dr. Eugene Allgower and Dr. Kurt Georg for the help and guidance during this study.

REFERENCES

[1] E. ALLGOWER, K. BÖHMER, K. GEORG, AND R. MIRANDA, *Exploiting symmetry in boundary element methods*, SIAM J. Numer. Anal., 29 (1992), pp. 534–552.

[2] E. ALLGOWER, K. GEORG, AND J. WALKER, *Exploiting symmetry in* 3-D *boundary element methods*, in Contributions in Numerical Mathematics, Vol. 2, R. Agarwal, ed., World Scientific Publ. Comp., Singapore, 1992, pp. 15–25.

[3] K. E. Atkinson, *Piecewise polynomial collocation for integral equations on surfaces in three dimensions*, J. Integral Equations Appl., 9 (1984), pp. 25–48.

[4] ———, *A survey of boundary integral equations for the numerical solution of Laplace's equation in three dimensions*, in Numerical Solution of Integral Equations, M. Goldberg, ed., Plenum Press, New York, 1990, pp. 1–34.

[5] K. E. Atkinson and I. G. Graham, *Iterative solution of linear systems arising from the boundary integral method*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 694–722.

[6] T. Chan, *An optimal circulant preconditioner for Toeplitz systems*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 766–771.

[7] P. Davis, *Circulant Matrices*, John Wiley and Sons, New York, 1979.

[8] J. D. Flores, *The conjugate gradient method for solving Fredholm integral equations of the second kind*, Internat. J. Computer Math., 48 (1993), pp. 77–94.

[9] K. Georg, *Approximation of integrals for boundary element methods*, SIAM J. Sci. Stat. Comput., 12 (1991), pp. 443–453.

[10] K. Georg and R. Miranda, *Exploiting symmetry in solving linear equations*, in International Series of Numerical Mathematics, Vol. 104, E. L. Allgower, K. Böhmer, and M. Golubitsky, eds., Birkhäuser Verlag, Basel, Switzerland, 1992.

[11] G. Golub and C. van Loan, *Matrix Computations*, Second ed., The Johns Hopkins University Press, Baltimore, London, 1989.

[12] W. Hackbusch, *Multigrid Methods and Applications*, Springer-Verlag, Berlin, 1985.

[13] ———, *Integralgleichungen*, Leitfäden Angew. Math. Mech., Vol. 68, 1989.

[14] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 105–126.

[15] J.-P. Serre, *Linear Representations of Finite Groups*, Graduate Texts in Mathematics, Vol. 42, Springer-Verlag, Berlin, Heidelberg, New York, 1977.

[16] J. Tausch, *A generalization of the discrete Fourier transformation*, in Exploiting Symmetry in Applied and Numerical Analysis, Lectures in Applied Mathematics, Vol. 29, E. L. Allgower, K. Georg, and R. Miranda, eds., American Mathematical Society, Providence, RI, 1993, pp. 405–412.

[17] ———, *Perturbation analysis for some linear boundary integral operators*, J. Integral Equations Appl., 3 (1995), to appear.

[18] E. E. Tyrtyshnikov, *Optimal and superoptimal circulant preconditioners*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 459–473.

[19] S. Vavasis, *Preconditioning for boundary integral equations*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 905–925.

[20] J. Walker, *Numerical experience with exploiting symmetry groups for boundary element methods*, in Exploiting Symmetry in Applied and Numerical Analysis, Lectures in Applied Mathematics, Vol. 29, E. Allgower, K. Georg, and R. Miranda, eds., American Mathematical Society, Providence, RI, 1993.

SIAM J. SCI. COMPUT.
Vol. 17, No. 1, pp. 100–117, January 1996

008

PERFORMANCE ISSUES FOR ITERATIVE SOLVERS IN DEVICE SIMULATION*

QING FAN†, P. A. FORSYTH‡, J. R. F. MCMACKEN§, AND WEI-PAI TANG‡

Abstract. Due to memory limitations, iterative methods have become the method of choice for large scale semiconductor device simulation. However, it is well known that these methods suffer from reliability problems. The linear systems that appear in numerical simulation of semiconductor devices are notoriously ill conditioned. In order to produce robust algorithms for practical problems, careful attention must be given to many implementation issues. This paper concentrates on strategies for developing robust preconditioners. In addition, effective data structures and convergence check issues are also discussed. These algorithms are compared with a standard direct sparse matrix solver on a variety of problems.

Key words. device simulation, iterative solver, preconditioning, ordering, drop tolerance, fill level

AMS subject classifications. 65F10, 78A35

1. Introduction. The increasing demands for more accurate semiconductor device modeling have pushed the development of numerical methods into a new era. One of these new developments is the study of techniques for the solutions of very large ill-conditioned sparse linear systems. In the past, software developers favored the use of sparse direct methods. These solvers could be treated as black box modules, since a reliable solution could (almost) always be obtained. However, the memory requirements of these direct methods make them impractical for large scale simulations. Consequently, attention has shifted to the use of iterative methods.

During the past decade, there has been considerable interest in Krylov subspace acceleration coupled with various preconditioning techniques. A great deal of effort [2], [26], [34], [36], [39], [19] has been devoted to developing iterative acceleration methods, while *robust* preconditioners have received less attention, mainly because of inherent theoretical difficulties. However, a good preconditioner is necessary for a robust iterative algorithm. Many workers have observed that a superior preconditioner can boost performance by an order of magnitude [5], [40], while a better acceleration technique may only improve the performance by 10–30%. In [5] and [34], various new developments in iterative methods for device simulation have been summarized.

Although iterative methods seem to be the only practical choice for large scale simulations, direct methods are still used in many situations. This is because many existing iterative methods may fail to converge when the matrix is very ill conditioned and careful attention is not given to many implementation issues.

The objective of this article is to investigate several issues which are important for the performance of iterative methods in device simulation. In particular, our emphasis will be on construction of an effective preconditioner. Both level-based and drop tolerance preconditioners [16] will be tested. A new two-step preconditioner, which treats the electric potential terms in the Jacobian in a very accurate manner, will also be developed. The alternate block factorization (ABF) technique, which will be shown to be equivalent to block scaling, has been suggested as a preconditioner for device simulation problems. The ABF preconditioner will

*Received by the editors May 13, 1994; accepted for publication (in revised form) February 21, 1995. This work was supported by the Natural Sciences and Engineering Research Council of Canada, and by the Information Technology Research Centre, which is funded by the Province of Ontario.

†Department of Applied Mathematics, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1 (q2fan@yoho.uwaterloo.ca).

‡Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1 (paforsyt@yoho.uwaterloo.ca) and (wptang@riacs.edu).

§Goal Electronics Inc., Waterloo, Ontario, Canada N2L 6J7 (jcmacke@mach1.wlu.ca).

be compared to the above methods. Some other issues which will also be addressed include the ordering of the unknowns [14], [16], [12], the choice of acceleration method, and the convergence check criteria. All these methods were tested in a commercially available drift-diffusion device simulator [31] which uses full Newton iteration for solution of the nonlinear algebraic equations.

2. CHORDV and test problems. CHORD System V [31] is a semiconductor device simulator which uses fully coupled Newton iteration to solve a variety of carrier transport models. In this paper, we are concerned with the traditional two-carrier, drift-diffusion equations: Poisson's equation and the electron and hole current continuity equations.

$$\nabla^2\psi + \frac{q}{\epsilon}(p - n + N_D - N_A) = 0,$$

$$\frac{\partial n}{\partial t} = \frac{1}{q}\nabla \cdot J_n - R,$$

$$-\frac{\partial p}{\partial t} = \frac{1}{q}\nabla \cdot J_p - R.$$

Here, ψ is the electrostatic potential, p, n the hole and electron concentrations, J_n, J_p the corresponding current densities, N_D, N_A the ionized donor and acceptor concentrations, and R the net recombination. Using the drift-diffusion approximation, we can write the electron and hole currents as

$$J_n = -q\mu_n n\nabla\psi + qD_n\nabla n,$$

$$J_p = -q\mu_p p\nabla\psi - qD_p\nabla p,$$

where μ_n, μ_p and D_n, D_p are the carrier mobility and diffusion coefficients. These expressions are combined to yield a system of three equations in three unknowns (ψ,n,p). Our carrier mobility models are taken from Nishida and Sah [32] and include components due to lattice vibration, ionized impurities, surface scattering, and velocity saturation. The recombination term includes Shockley–Read–Hall, Auger, and impact ionization. We convert the diffusion coefficients to effective mobilities using the Einstein relation.

The three equations are discretized across a two-dimensional (2D) domain using box integration [35] applied to a nonorthogonal grid. Box integration can also be viewed as a finite volume-type method [3]. In the finite volume approach, the differential equations are integrated over finite volumes, and the volume integrals are converted to surface integrals. Consider the grid node i and its neighbors j. We begin by constructing perpendicular bisectors of the grid to develop a control volume. Note that this approach restricts us to using grid meshes which are Delauney triangulations.

Next we assume that the electric field and current are constant across each face of the polygon as well as all physical properties. Using a finite volume method, the discrete Poisson equation is

$$\sum_j \left[\frac{h_j^+ + h_j^-}{l_j}\right][\psi_j - \psi_i] + \frac{q}{\epsilon}[p_i - n_i + N_{Di} - N_{Ai}]\sum_j \left[a_j^+ + a_j^-\right] = 0$$

where l_j is the distance from node i to node j (see Fig. 2.1).

Backward Euler timestepping is used, and consequently, the discrete electron continuity equation becomes

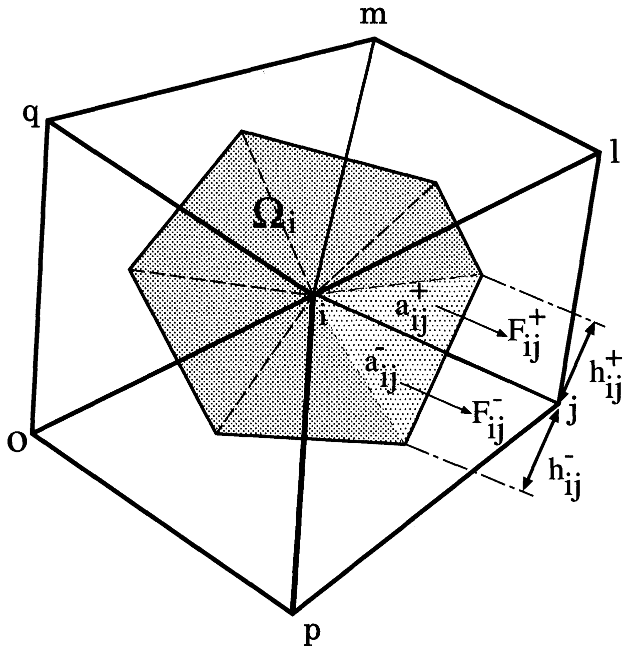


FIG. 2.1. *Box integration scheme for discretization.*

$$\frac{1}{q}\sum_j \left[h_j^+ + h_j^-\right][J_n \cdot \hat{\eta}_{ij}]_j^{k+1} - \left[[R_i]^{k+1} + \frac{n_i^{k+1} - n_i^k}{\Delta t}\right]\sum_j \left[a_j^+ + a_j^-\right] = 0$$

where $\hat{\eta}_{ij}$ is the outward pointing normal to Ω_i along the face between node i and node j. The Scharfetter–Gummel approximation for current density [37] is given by

$$[J_n \cdot \hat{\eta}_{ij}]_j^{k+1} = -q\frac{\phi_T \mu_n}{l_j}\left[n_i^{k+1}B(-\delta\psi_j^{k+1}/\phi_T) - n_j^{k+1}B(\delta\psi_j^{k+1}/\phi_T)\right]$$

where

$$B(x) = \frac{x}{\exp x - 1},$$

$$\delta\psi_j = \psi_j - \psi_i,$$

and ϕ_T is the thermal voltage. Thus, the resulting discrete form of the electron continuity equation is

$$\frac{1}{q}\sum_j \left[h_j^+ + h_j^-\right] - q\frac{\phi_T \mu_n}{l_j}\left[n_i^{k+1}B(-\delta\psi_j^{k+1}/\phi_T) - n_j^{k+1}B(\delta\psi_j^{k+1}/\phi_T)\right]$$

$$-\left[[R]_i^{k+1} + \frac{n_i^{k+1} - n_i^k}{\Delta t}\right]\sum_j \left[a_j^+ + a_j^-\right] = 0.$$

A similar expression may be developed for the hole continuity equation.

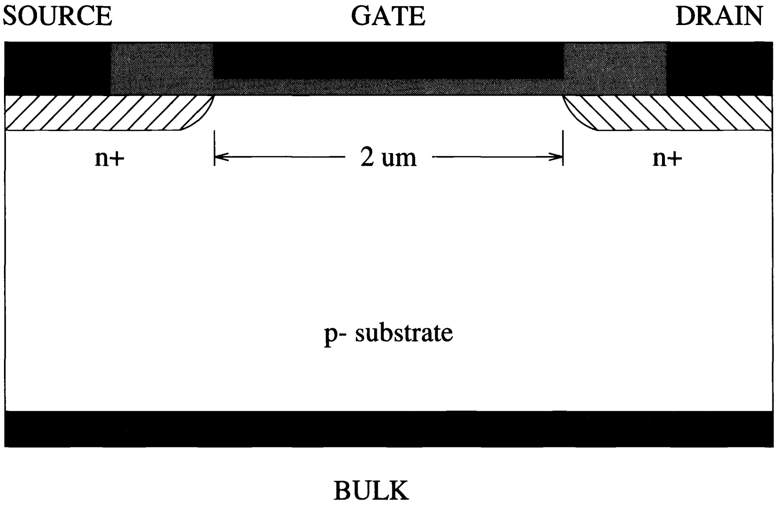


FIG. 2.2. *An n-channel MOSFET.*

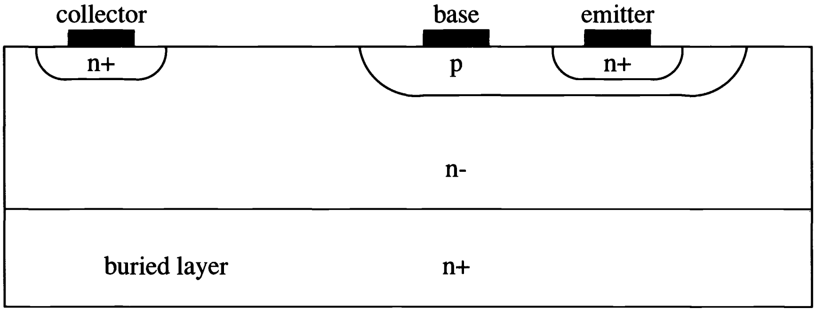


FIG. 2.3. *A bipolar junction transistor.*

In CHORD, the transport model is solved using Newton iteration. Given a set of equations with residuals r_k, associated with unknowns x_k, we linearize the system about a point x_k and solve the linear system $J_k(x_{k+1} - x_k) = J_k \Delta x_k = -r(x_k)$ where J_k is the Jacobian matrix formed by computing partial derivatives of r_k. In our case, x_k is typically the vector of unknowns (ψ, n, p). Our solution estimate is then updated by the Newton step Δx_k and the process repeated until the system is converged. In this paper, we will focus on iterative methods for solving the linear system.

Two typical semiconductor models, metal-oxide-semiconductor field-effect transistor (MOSFET) and bipolar junction transistor (BJT), are used for testing purposes. The n-channel MOSFET device is a simplified self-aligned n-channel MOSFET with a 2um drawn channel length and a 25nm thick gate insulator (see Fig. 2.2). The source and drain are 0.25um abrupt junctions in a lightly doped p-type substrate ($5.0e + 15$ cm^{-3}). There is no channel implant. We assume an ideal structure with no oxide charge or interface charge. The MOSFET problem has 15587 unknowns.

The second device is a bipolar junction transistor (see Fig. 2.3) that is an active three-terminal device which can be used as an amplifier or switch. There are areas of applications in which the bipolar transistor is superior to MOSFET, such as in high-power devices and in high-speed logics for high-performance computers. This device is a simple vertical *npn* transistor formed by two ion-implant steps and a thermal anneal, an $n+$ buried layer is used to reduce the parasitic collector resistance. The BJT problem has 13758 unknowns.

Three sets of test problems were developed based on the two device structures. In the MOSA test set, the MOSFET source and substrate were grounded and the drain held at 5.0V while the gate bias was swept from 0.0V to 10.0V in 0.5V increments. In MOSB, the gate was fixed at 1.0V while the drain was swept from 0.0V to 10.0V, again in 0.5V steps. In the final problem, BJT, a single bias point (zero on emitter, base, and collector) was applied to the bipolar junction transistor.

Each of these subproblems requires many Newton steps, which in turn requires many Newton solves. Consequently, each test problem exercises the matrix solution algorithm over many different Newton iterations and values of the source-drain voltage. Consequently, the total time for the entire test problem is a good indication of the robustness and reliability of the matrix solver over a wide range of situations.

The Newton iteration convergence criteria used in these problems was a relative one. Given the Newton step Δx_i and solution estimate x_i, the iteration was stopped when the relative error $|\Delta x_i / x_i|$ was less than 0.001 for all variables. To avoid numerical inaccuracies, this test was skipped if $|x_i| < 10^{-7}$. This situation is most often found with minority carrier concentrations. For our problems, values of this magnitude do not contribute to the solution.

3. Inner iteration convergence criteria. The complete solution process in a device simulation consists of an inner and an outer iteration. The outer iteration is the nonlinear Newton method. If the Jacobian is solved using an iterative method, then the convergence tolerance must be specified. For the first few Newton steps, the nonlinear residual in (4.2) is quite large. Therefore, a large residual reduction in the linear iteration is needed for an accurate Δx (where x is the vector of unknowns). After the Newton iteration reaches a certain accuracy, the nonlinear residual becomes smaller. The update Δx only affects the last few digits of the final solution. As a result, the relative residual reduction requirement for the solution of the Jacobian can be lowered.

Although at first sight this seems to be contrary to the usual approach suggested in, for example, [18], where a loose tolerance is used at the initial stages of the Newton iteration, and a tighter tolerance is used at the final stages, this can be understood as follows. An inner iteration convergence tolerance of reducing the l_2 residual by 10^{-3} is sufficient, at least in our experience, to ensure quadratic convergence of the Newton iteration, as convergence is approached. Therefore, this tolerance is to be considered quite tight. However, we have found that a loose tolerance for the initial Newton iterations, which are far away from the solution, often results in divergence of the Newton iteration. This problem can sometimes be cured by using an even tighter inner iteration tolerance for the initial Newton iterations. A similar effect has been observed when using Newton iteration for the incompressible Navier–Stokes equations [11]. In the fluid dynamics case, it can be argued that a loose tolerance will result in large mass balance errors (the mass balance equation is linear for incompressible flow), and hence makes the iteration unstable. We suspect that in the device simulation case, the electric potential equation must be solved quite accurately, or else the iteration will tend to diverge.

In this work, a dynamic switch of the linear convergence criteria is implemented in the linear solver. Initially, the convergence requirement is that the initial l_2 residual be reduced by 10^{-6}. After the *nonlinear* residual is reduced below 10^{-3} (compared to the initial *nonlinear* residual), the linear residual reduction switches to 10^{-3}. This dynamic switch can generally save 10–15% of the CPU time.

4. Block data structure and scaling. Since device simulation involves a system of coupled partial differential equations, it is natural to exploit the block structure of the Jacobian in order to obtain a good preconditioner. Two methods that use this concept are the ABF [4] and the modified singular perturbation (MSP) [41].

For the purposes of illustration, assume that the device model in question is a drift-diffusion model having three coupled partial differential equations. If the Jacobian equations are ordered so that all the electric potential equations are grouped first, followed by the electron conservation equations, and then the hole conservation equations, and the unknowns are ordered so that all the electric potentials are first, then the electron concentrations, and finally the hole concentrations, then the Jacobian matrix can be partitioned as

$$J = \begin{bmatrix} J_{\phi\phi} & J_{\phi n} & J_{\phi p} \\ J_{n\phi} & J_{nn} & J_{np} \\ J_{p\phi} & J_{pn} & J_{pp} \end{bmatrix} \tag{4.1}$$

where J_{ij}, $i, j = \phi, n, p$ denotes the block of derivatives of the equation for electric potential, electron conservation, or hole conservation, with respect to the electric potential, electron concentration, or hole concentration. Then the ABF preconditioner is

$$D^{-1} = \begin{bmatrix} D_{\phi\phi} & D_{\phi n} & D_{\phi p} \\ D_{n\phi} & D_{nn} & D_{np} \\ D_{p\phi} & D_{pn} & D_{pp} \end{bmatrix}^{-1}$$

where D_{ij} is the diagonal matrix of J_{ij}. More recently, the approximate block elimination (ABE) has been proposed which uses an incomplete 3×3 block factorization of the original block structured Jacobian [40].

In this paper, another kind of block structure of the Jacobian matrix is used. The unknown variables at each physical grid node are grouped together to form an $m \times m$ block matrix, where m is the number of grid nodes. In general, the diagonal block elements of this block Jacobian may have different sizes since the number of unknowns for each node varies. This is due to the fact that different models are used in different device materials and at the device contacts. The motivation for explicitly considering the block structure is the following.

All the nonzero blocks are regarded as dense. Consequently, we need only to specify the sparsity pattern for the nonzero blocks. As a result, the integer space needed for specifying the structure of the block sparse matrix is an order of magnitude less compared to the original scalar sparse matrix (assuming that the average number of unknowns per node is at least three). This block structure will be more attractive when the semiconductor model becomes more sophisticated, since typically, more detailed physics requires more equations per node. Our study [9] indicates that one of the basic operations in iterative methods, namely matrix vector multiplication, takes less time if a block data structure is used compared to the scalar case. It is clear that the cache hit ratio will be higher in the block case.

More importantly, the new block structure can allow us to reduce the strong coupling between unknowns associated with a single grid node before the iterative process and other preconditioning steps begin. If we order the unknowns and equations so that all equations and unknowns associated with a node are ordered consecutively, then

$$J = \begin{bmatrix} J_{11} & J_{12} & \cdots & J_{1n} \\ J_{21} & J_{22} & \cdots & J_{2n} \\ & & \ddots & \\ J_{n1} & J_{n2} & \cdots & J_{nn} \end{bmatrix}$$

where J_{ii} are typically $1 \times 1, 2 \times 2, \ldots, 7 \times 7$ block matrices. For drift-diffusion models, the number of unknowns in each block may vary. Usually, there are three unknowns (ψ, n, p), but if the computational domain contains an oxide, then only the electric potential equation

TABLE 4.1
Block scaling at ground bias point, where N is the number of unknowns.

Test (N)	Methods applied	Total linear iterations	Total linear CPU time
MOSA	ABF	1723	1697.67
(15583)	Two step	73	465.04
BJT	ABF	Not converged	
(13758)	Two step	117	523.54

is solved (with ψ as an unknown) in the oxide region, since it is assumed that there are no charge carriers in an oxide. Some of the examples in this work contain oxides. In addition, at contacts, it is sometimes convenient to impose an additional boundary condition in terms of the total current. This total current is added to the list of unknowns at a contact node. However, in this paper, none of the examples have this additional electric current unknown. The maximum size of J_{ii} is then usually four, with minimum size being 1×1. Recall that at each Newton step we need to solve the equation

$$J\Delta x = -r \tag{4.2}$$

where J is the Jacobian, Δx is the vector of updates, and r is the residual vector.

First, a block scaling is applied to equation (4.2). The scaling matrix

$$S = \begin{bmatrix} J_{11}^{-1} & 0 & \cdots & 0 \\ 0 & J_{22}^{-1} & \cdots & 0 \\ & & \ddots & \\ 0 & 0 & \cdots & J_{nn}^{-1} \end{bmatrix}$$

is multiplied on both sides of equation (4.2). The preconditioned Krylov space method is then applied to the scaled matrix equation

$$SJ\Delta x = Sr.$$

Note that this scaling step is equivalent to applying the ABF preconditioner if a permutation is applied. The difference here is that a further preconditioning process will be applied to the scaled system. The improvement in using a block scaling followed by further preconditioning is significant compared to using block scaling alone.

Table 4.1 shows the difference in performance if a further preconditioning step is taken. Two Jacobian matrices were generated at intermediate Newton iterations from MOSA and BJT. The two-step preconditioner uses a block scaling followed by the best preconditioned BI-CGSTAB method which will be described in more detail in following sections. Table 4.1 shows the total number of solver iterations for all Newton iterations, as well as the total number of unknowns N. Clearly, block scaling (ABF) alone is not sufficient for a robust technique.

5. Preconditioning issues. Preconditioned Krylov subspace methods are the standard iterative methods for device simulation. However, much of the past research has concentrated on the behavior of different acceleration methods [1], [5], [7], [22], [25], [28], [29], [34]. Many authors have observed that GMRES, CGS, and Bi-CGSTAB are the relatively robust choices among the many. Both CGS and Bi-CGSTAB are BCG-type methods, but are now more widely used than the original BCG algorithm. Bi-CGSTAB seems to be a good choice

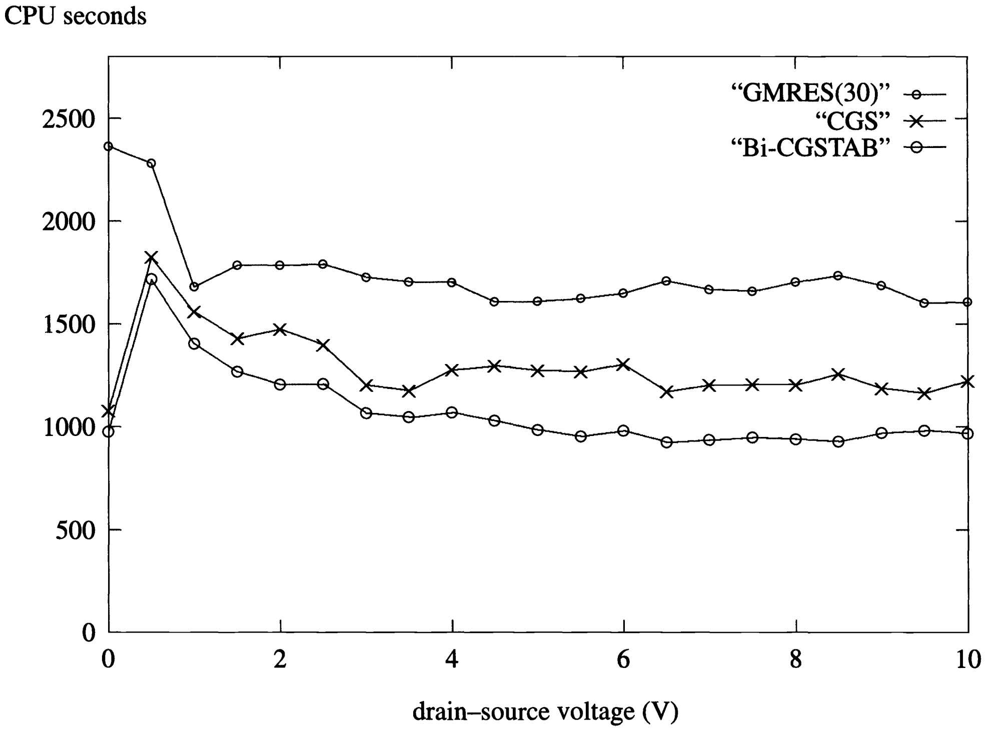


FIG. 5.1. *Comparison of various acceleration methods* (5.1).

from among the BCG methods, since convergence is rapid and less prone to the wild residual oscillations of CGS. QMR [26] is a newly developed Krylov subspace method which has very smooth convergence behavior. Thusfar, we have carried out only limited testing using the QMR method, and it is too soon to draw firm conclusions about the performance of QMR compared to Bi-CGSTAB or GMRES.

In particular, we found that Bi-CGSTAB is generally the most robust method. For example, Fig. 5.1 shows the total CPU times for the MOSB problem with an $ILU(1)$ preconditioner at various values of the drain-source voltage. Clearly, Bi-CGSTAB is the most efficient acceleration method. Tests for other problems showed a similar trend. Consequently, all computations will be carried out using Bi-CGSTAB for the remainder of this paper.

For completeness, we give the preconditioned Bi-CGSTAB algorithm below.

(LU represents the incomplete factorization of matrix A, and $\overline{Aq}$ is a vector.)

$$\begin{array}{rcl}
r_0 & = & b - Ax_0 \\
\hat{\omega}_0 & = & \beta = \alpha_0 = 1 \\
\overline{Aq}_0 & = & q_0 = 0 \\
\textbf{For} & & i = 1, 2, \ldots \\
 & & \hat{\beta} = (r_0, r_{i-1}); \quad \omega_i = (\hat{\beta}/\beta)(\bar{\omega}_{i-1}/\alpha_{i-1}) \\
 & & \beta = \hat{\beta} \\
 & & q_i = r_{i-1} + \omega_i(q_{i-1} - \alpha_{i-1}\overline{Aq}_{i-1}) \\
 & & \bar{q}_i = (LU)^{-1} q_i
\end{array}$$

```
        $\overline{Aq}_i = A\bar{q}_i$
        $\hat{\omega}_i = \hat{\beta}/(r_0, \overline{Aq_i})$
        $s = r_{i-1} - \hat{\omega}_i \overline{Aq}_i$
        $\bar{s} = (LU)^{-1}s$
        $t = A\bar{s}$
        $\alpha_i = (L^{-1}t, L^{-1}s)/(L^{-1}t, L^{-1}t)$
        $x_i = x_{i-1} + \hat{\omega}\bar{q}_i + \alpha_i \bar{s}$
        if $x_i$ accurate enough, then quit
        $r_i = s - \alpha_i t$
    End
```

In this paper, we will concentrate on the issues related to incomplete LU preconditioners, since these are regarded as among the most robust for semiconductor simulations [5], [29].

Before we pursue the different issues in constructing a good preconditioner, a block graph representation of the Jacobian is useful. If we view the grid node v_i as the node of a graph representing the block sparse matrix and map the nonzero block element J_{ij} to an edge connecting the two nodes v_i, v_j, the block sparse Jacobian

$$J = \begin{bmatrix} J_{11} & J_{12} & \cdots & J_{1n} \\ J_{21} & J_{22} & \cdots & J_{2n} \\ & & \ddots & \\ J_{n1} & J_{n2} & \cdots & J_{nn} \end{bmatrix} \tag{5.1}$$

can be represented by a graph form. During the factorization process, new nonzero block elements will be introduced. A notion of "fill level" can be introduced to each position of the block matrix. We initially define

$$\text{level}_{ij}^{(0)} = \begin{cases} 0, & \text{if } J_{ij} \neq 0, \\ \infty, & \text{otherwise.} \end{cases} \tag{5.2}$$

The nonzero block elements then have a fill level 0. As the elimination proceeds, fill will be introduced. At the kth step of the elimination, the fill levels are given by [16]

$$\text{level}_{ij}^{(k)} := \min\left(\text{level}_{ik}^{(k-1)} + \text{level}_{kj}^{(k-1)} + 1, \text{level}_{ij}^{(k-1)}\right).$$

In other words, the fill level of a fill is the length of the shortest path between nodes v_i and v_j through the nodes eliminated before v_i and v_j [27]. The fill level is commonly used to decide the sparsity pattern of an ILU factorization.

5.1. Ordering. The matrix ordering affects the computational efficiency of a matrix solver in many ways. For direct methods, a good ordering technique is essential in order to minimize the amount of fill. In parallel (or vector) processing, ordering again plays a crucial rule. A number of studies have examined the effect of matrix ordering on the quality of preconditioners for iterative methods based on an incomplete factorization [8], [14], [16], [23], [24], [33], [21]. In [14], [16], and [17] evidence was presented to demonstrate that matrix ordering can have a profound effect on the quality of preconditioners. A heuristic method was developed that was shown to produce good matrix ordering. Unfortunately, some of the techniques for an effective ordering developed in [14], [16], and [17] can only be applied to scalar sparse matrices. The ordering generated by these new algorithms may destroy the block pattern we employed here. However, the graph-based orderings (where we view each block of the Jacobian as a node in the graph) can handle the block matrix (5.1) naturally.

The algorithm for generating a scalar ILU preconditioner can be directly extended to the block case. The block incomplete LU factorization can then be interpreted as a graph elimination process [27]. The graph representation of (5.1) is in fact the graph of the grid used to discretize the device. Note that most of the grid generation algorithms in semiconductor simulation involve some kind of refinement process. Consequently, the ordering of the grid nodes can be very scattered. In addition, after the refinement the resulting grid is usually unstructured. Consequently, there is no obvious natural way to order the grid nodes. The reverse Cuthill–McKee (RCM) ordering [27] originally was proposed as a good technique for reducing the profile of a sparse matrix. The basic idea of this ordering algorithm is to construct the level set from a starting node of the graph representing the sparse matrix. The reverse order of the level set will produce a small profile. We can also view the RCM ordering as a generalized natural ordering of an unstructured grid. For a rectangular grid, RCM ordering is the diagonal ordering of the mesh, if a corner node is chosen as the starting node. This ordering algorithm can be generalized to the block Jacobian case, in an obvious way.

RCM ordering is known as an effective ordering [23], [24] for ILU preconditioning in many applications. The standard RCM ordering can produce a poor ordering (for an iterative solver) if the problem is anisotropic. A revised version for a weighted graph can alleviate this problem [12]. A comparison of an ILU method which uses the original ordering and the RCM ordering was carried out for MOSA test set. Figure 5.2 gives the total CPU time required to obtain the steady state solution for various values of the drain-source voltage for MOSA, using the original ordering (ORI), RCM ordering (RCM), and, for comparison, the time for a direct solver (SPARSPAK) [10] is also shown. Here an ILU(0) preconditioner is used with Bi-CGSTAB acceleration. On average, RCM ordering reduces the total CPU time by about 30–40% compared to the original ordering. Consequently, in the following, RCM ordering will be used unless otherwise noted.

5.2. Sparsity pattern. The key step during the incomplete LU factorization process is to determine the sparsity pattern of the L and U. To find the optimal sparsity pattern for the ILU factorization is a much more difficult task than the solution of the Jacobian itself. Typically, some simple heuristics are used to determine if a fill element will be discarded. The common strategies are the following.

1. By a drop tolerance, $ILU(\varepsilon)$. A drop tolerance method discards numerically "small" values of fills. There are many possible drop tolerance criteria [16]. Let $a_{ij}^{(0)}$ be the (scalar) entry of the original Jacobian matrix and $a_{ij}^{(k)}$ be the element of the submatrix that remains after $(k-1)$ steps of the incomplete factorization. A possible drop tolerance method is

$$|a_{ij}^{(k)}| < \varepsilon\sqrt{|a_{ii}^{(0)}a_{jj}^{(0)}|}. \tag{5.3}$$

For symmetric systems, this is the same criteria as in [42]. Note that since the Jacobian matrix is not symmetric positive definite, we do not use the diagonal modification suggested in [30]. Even for symmetric positive definite problems, the diagonal modification usually results in a slow method [20].

It is probably not a good idea to extend the drop tolerance approach to the block case. The elements in each of the small block matrices of the Jacobian in device simulation often vary by many orders of magnitude ($10^{10} - 10^{16}$). One large entry in a block element may result

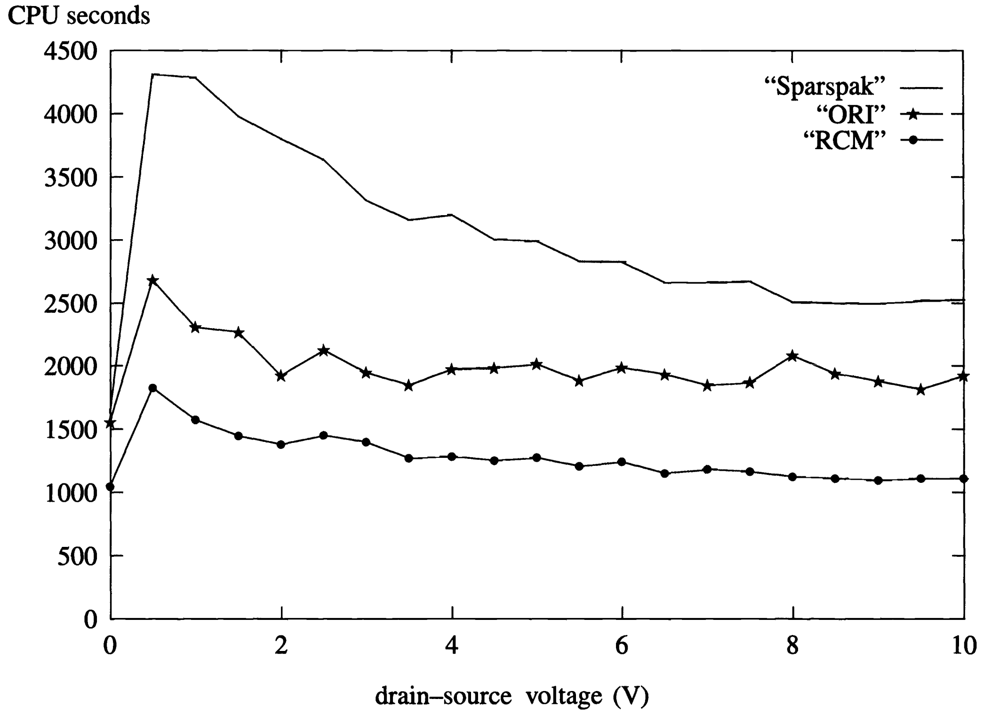


FIG. 5.2. *Comparison of direct solver (Sparspak) and iterative solver, RCM ordering (RCM) and ORI ordering (ORI). ILU(0) preconditioner and Bi-CGSTAB acceleration were used in the iterative solver.*

in keeping the entire fill block, which may not be desirable. Consequently, the drop tolerance method will be applied to each individual element in the Jacobian matrix in the following. A similar approach was used in [42].

In general, we have found that a drop tolerance incomplete factorization is not an effective technique for semiconductor device simulations. The characteristics of the Jacobian in this particular application cause problems for a drop tolerance approach. As we mentioned earlier, the elements in the Jacobian matrix can differ by a factor of 10^{14}. Several variables with very different scales coexist in the same submatrix. A small element may not imply insignificant influence on the solution. This effect has also been seen when applying a drop tolerance incomplete factorization to incompressible Navier–Stokes problems [13]. In this case, the drop tolerance worked well for small Reynolds numbers, but was very poor for high Reynolds numbers. This indicates that at high Reynolds numbers, the fluid flow problem is inherently poorly scaled, and so it becomes difficult to determine appropriate drop tolerance criteria. We believe that a similar effect occurs in device simulation. Of course, since the Jacobian itself is very ill conditioned, we can expect that dropping terms in an incomplete factorization (whatever drop strategy is used) may cause large changes in the solution, and hence result in a poor preconditioner.

The drop tolerance method (applied with criteria (5.3) to each element of the Jacobian) was compared with use of a two-stage method (to be described in the following section). Table 5.1 shows the total CPU time required for the matrix solve, the number of iterations, and the storage required (for a single value of the source-drain voltage), for various values of the drop tolerance ε. Both incomplete factorization and preconditioning cost are increasing as ε decreases.

TABLE 5.1

Drop tolerance compared to two step. Device MOSFET is tested at one representative bias point from MOSB test set. Device BJT is tested at the ground state.

Test (N)	Methods applied	Total linear iterations	Total linear CPU time	Kilo bytes
MOSB	Two step	131	820.52	8702
	$\varepsilon = 0.5$	1099	3195.84	5281
(15583)	$\varepsilon = 0.1$	999	3638.57	5838
	$\varepsilon = 0.01$	570	3951.03	9130
	$\varepsilon = 0.001$	391	6273.49	14273
BJT	Two step	117	523.54	7897
	$\varepsilon = 0.5$	1439	3700.06	4280
(13758)	$\varepsilon = 0.1$	1226	3320.64	4580
	$\varepsilon = 0.01$	1029	3505.31	6177
	$\varepsilon = 0.001$	579	2980.67	8867

Table 5.1 indicates that decreasing ε further for the MOSB problem will result in an increase in total CPU cost due to the expensive incomplete factorization stage. For problem BJT, the total CPU cost is still decreasing as ε decreases, but the final entry in Table 5.1 ($\varepsilon = .001$) indicates that the storage required for the drop tolerance incomplete factorization is larger than the storage required for the two-step method, and the drop tolerance approach is more than five times slower than the two-step method.

In general, the drop tolerance preconditioner is between six and ten times slower than the two-step preconditioner. It is clear that, for the same amount of storage, the two-step method is far superior to the drop tolerance approach.

2. By fill level, $ILU(\ell)$. The fill block $J_{ij}^{(k)}$ at the kth step of the factorization will be discarded if

$$\text{level}_{ij}^{(k)} > \ell.$$

It is clear that the larger the level ℓ the better the preconditioner. When $\ell = n$, a complete factorization is obtained. However, large values of ℓ will be too expensive in terms of storage for 3D problems. Our experiments indicate that the performance (in terms of total CPU time) stops improving after $\ell = 2$ even for 2D problems.

For the level approach, only one symbolic factorization is needed for the entire simulation as long as the grid does not change. The sparsity patterns of the incomplete factorization are the same for different Newton steps or timesteps. As a result, the numerical and symbolic factorization can be separated to make the factorization process more efficient. This contrasts with the drop tolerance incomplete factorization, since a different sparsity pattern will result when the Jacobian changes. Therefore, the incomplete factorization process is more expensive for a drop tolerance preconditioner. The level approach is not only easy to implement, it is even more efficient for the block Jacobian case. Indeed, the symbolic factorization phase in this case costs only a very small fraction of symbolic factorization cost if the Jacobian was not considered as a block matrix.

Table 5.2 shows the total CPU times for the MOSA test set at various levels of fill of the ILU. The improvement from level zero to level one is significant. However, the improvement in going to levels higher than one is marginal. We also list a detailed record of a typical simulation in Fig. 5.3 (for a single value of the source-drain voltage). We can see that the number of iterations is monotone decreasing as the level increases. However, the amount of fill for the preconditioner becomes larger as the level of fill increases. Therefore, the higher cost for each iteration will eventually outweigh the reduction in number of iterations. Note

TABLE 5.2
Comparison of levels 0, 1, 2, 3, *and* 4, *MOSA test.*

Method	Newton iter #	Linear iter #	Total CPU linear time	Total CPU time	Kilo bytes
Level(0)	13	178	701.33	898.18	6648
Level(1)	12	111	510.02	681.79	7391
Level(2)	12	85	485.42	670.90	8702
Level(3)	12	83	558.10	729.52	10116
Level(4)	12	70	580.68	768.87	11498
Sparspak	12	12	1658.10	1852.57	20234

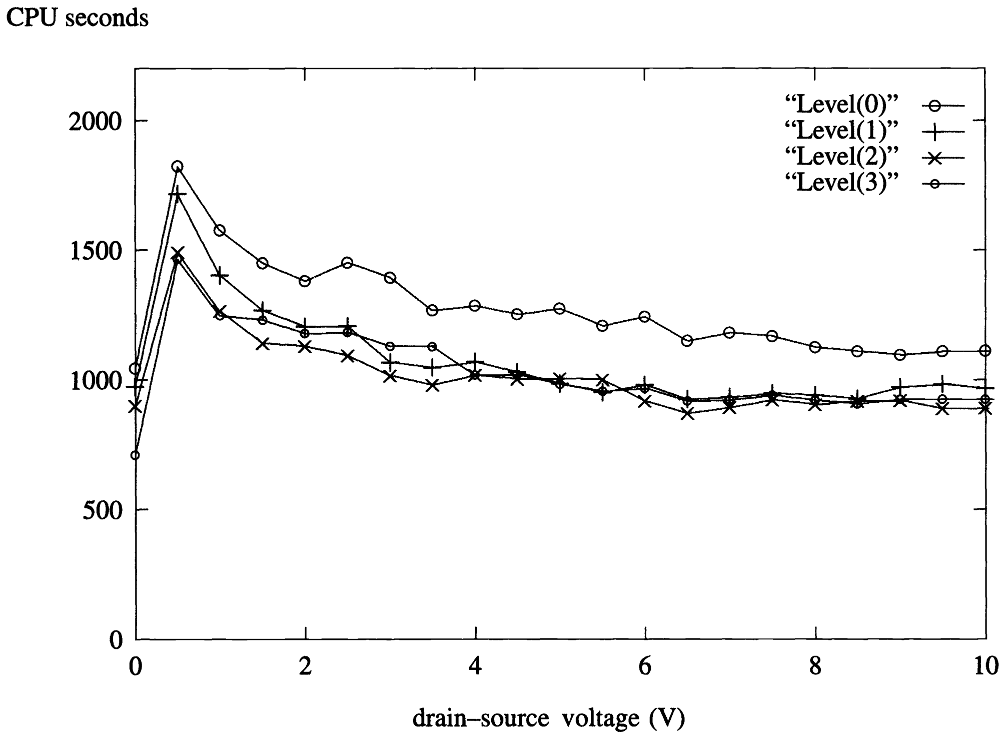


FIG. 5.3. *Comparison of levels* 0, 1, 2, *and* 3 *using one-step preconditioner in Bi-CGSTAB acceleration and RCM ordering, MOSA test.*

from Table 5.2 the level(0) test requires one more Newton iteration than other tests. This is because the Jacobian iterations are not precisely the same for all the tests, due to the finite convergence tolerance for the inner iteration. In the case of the level(0) tests, this results in an extra Newton iteration before the nonlinear convergence test is satisfied.

3. **By the combination of both,** $ILU(\ell, \varepsilon)$. A fill entry is dropped if

$$\text{level}^{(k)}_{ij} > \ell \quad \text{or} \quad |J_{ij}| < \varepsilon.$$

This is a useful heuristic for many applications [16]. However, since the drop tolerance approach by itself does not appear to be very useful in this application, we will not pursue this method further.

5.3. Two-step preconditioner. As we can see from the experiments of the previous sections, when the accuracy of the factorization (by using a higher level fill or a smaller drop tolerance) improves, the number of iterations required for convergence decreases. However, the improvement in the number of iterations will not compensate for the higher cost of each iteration after a certain point. The best combination of the strategies thus far is to use RCM ordering plus ILU(1) or ILU(2). For a 3D problem, the ILU(2) approach may require too much storage. In the following, we will use an ILU(1) factorization unless otherwise noted.

Let L_J and U_J be the incomplete factorization of the Jacobian matrix J. The techniques used in the previous sections attempt to reduce the difference

$$\|J - L_J U_J\|.$$

The basic assumption of this method is that if this difference is small, then the solution y of the preconditioning step

$$L_J U_J y = p \tag{5.4}$$

will be close to the solution of

$$Jx = p.$$

Alternatively, we can use the following criteria for producing an effective preconditioner. Denote M_j a preconditioner of J and y the solution of

$$M_j y = p.$$

A better preconditioner M_j means a smaller residual of

$$r = Jy - p. \tag{5.5}$$

When $r = 0$, a perfect preconditioner is obtained. Therefore a refined preconditioner or a two-step preconditioner can be developed as follows.

For the purposes of illustration, assume that the variables are ordered as in (4.1). After we solve the equation (5.4), the residual r in (5.5) is calculated. Let $J_{\phi\phi}$ be the diagonal block element for the potential variable in (4.1) and r_ϕ be the part of residual in (5.5) corresponding to the electric potential equations. The equation

$$J_{\phi\phi} \Delta y_\phi = r_\phi \tag{5.6}$$

is solved. This problem is much smaller and better conditioned. Let $\Delta y = [\Delta y_\phi, 0, 0]^T$, i.e., only the potential variables become updated. Then, the refined solution $\tilde{y} = y - \Delta y$ is used as the solution for the new two-step preconditioner M_t such that

$$M_t \tilde{y} = p.$$

It is easy to see that the new residual

$$\tilde{r} = J\tilde{y} - p$$

will have

$$\tilde{r}_\phi = 0.$$

TABLE 5.3

Comparison of one-step and two-step preconditioners with Bi-CGSTAB acceleration at two representative bias points from MOSA. Here (1:1) *indicates ILU*(1) *and* (1:2) *indicates ILU*(1) *and two-step preconditioner.*

Test situation	Bi-CGSTAB (level:step)	Newton iter #	Linear iter #	Total CPU linear time	Total CPU time
Bias point 1	Bi-CGSTAB(1:1)	20	459	1121.90	1360.44
	Bi-CGSTAB(1:2)	20	260	1114.53	1346.76
	SPARSPAK	20		2720.84	2959.65
Bias point 2	Bi-CGSTAB(1:1)	31	1468	3152.43	3507.59
	Bi-CGSTAB(1:2)	31	729	2741.86	3107.96
	SPARSPAK	31		4295.57	4656.91

TABLE 5.4

A performance comparison between different implementations. Device MOSFET is tested at one representative bias point from MOSB. Device BJT is tested at the ground state.

Test (N)	Methods applied	Linear iter #	Total CPU linear time	Total CPU time	Kilo bytes
MOSB (15583)	Two-step*	217	1081.55	1373.43	7391
	ILU(2)	352	1165.90	1460.94	5930
	ILU(1)	483	1227.65	1514.86	4618
	ABF		not converged		3875
	ILU(0.5)$^+$	1672	4794.89	5111.91	5281
	ILU(0.1)$^+$	1417	5193.31	5502.08	5838
	ILU(0.01)$^+$	863	5734.76	6057.82	9130
	ILU(0.001)$^+$	666	9217.64	9536.28	14273
BJT (13758)	Two-step*	142	544.58	696.00	6699
	ILU(2)	242	612.72	764.00	5384
	ILU(1)	322	652.69	827.00	4186
	ABF		not converged		3498
	ILU(0.5)$^+$	1439	3700.06	3847.00	4280
	ILU(0.1)$^+$	1226	3320.64	3465.00	4580
	ILU(0.01)$^+$	1029	3505.31	3645.00	6177
	ILU(0.001)$^+$	579	2980.67	3123.00	8867

* Two-step preconditioner with ILU(1).

$^+$ Drop tolerance ILU(ε), where ε = drop tolerance.

In other words, the two-step preconditioner has more accurate electric potential solution. Consider a case where the drift flow of holes and electrons dominates the diffusion flux. In this case, as the mesh size is reduced, the electric potential derivatives in the Jacobian will dominate the hole and electron concentration derivatives. Essentially, this is because in this situation, the electric potential is an elliptic-type variable, while the hole and electron concentrations are hyperbolic-type variables.

For the range of 2D problems we have tested so far, we have found that use of a direct solve of equation (5.6) is quite efficient. In other words, $J_{\phi\phi}$ is factored once, and equation (5.6) is solved by a forward and back solve each iteration. We use minimum degree ordering [10] for the initial complete factorization of $J_{\phi\phi}$ (this system is much smaller than the original Jacobian). Of course, for larger 3D problems, it may be more efficient to use an iterative method to solve equation (5.6).

Note that the two-step method is similar to the combinative technique used in [6]. Tables 5.3 and 5.4 present some detailed comparisons between the one-step and two-step precondi-

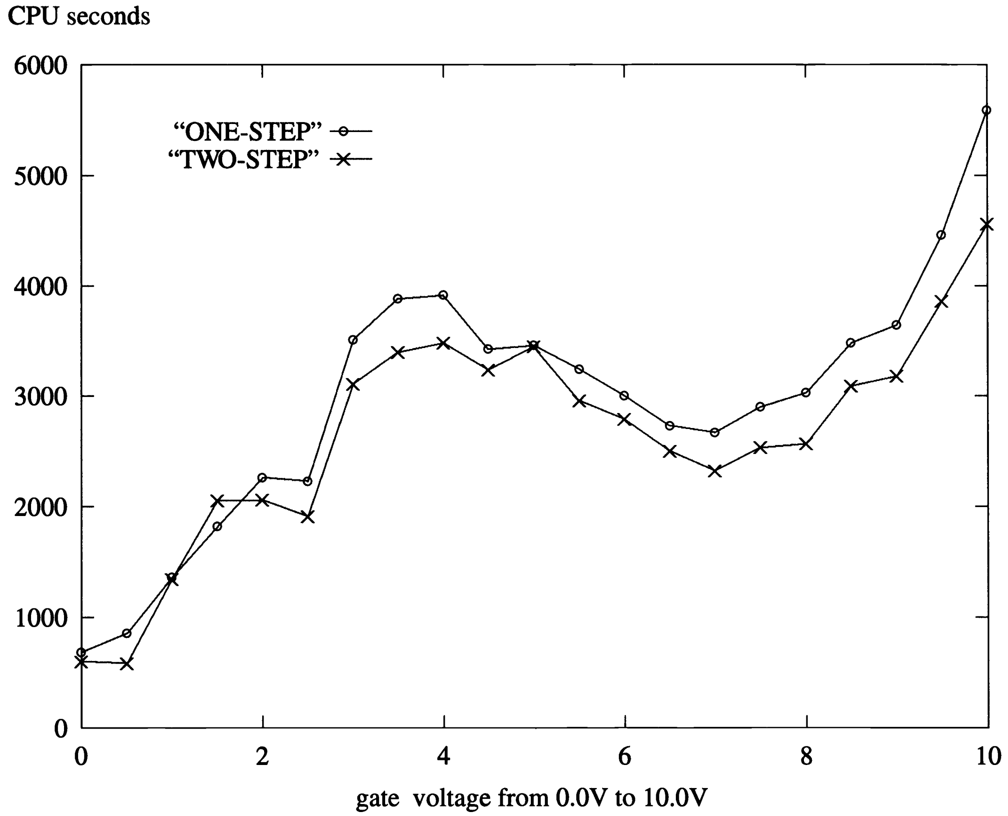


FIG. 5.4. *Comparison of one-step and two-step preconditioners.*

tioners for MOSA, MOSB, and BJT. We can see that the total number of linear iterations is reduced significantly with the two-step preconditioner. Figure 5.4 lists the CPU time comparison for a complete test run for test MOSA. The ILU(1) with Bi-CGSTAB acceleration is used for both one-step and two-step preconditioners. Block RCM ordering is applied. Of course, the new preconditioner is more expensive than the single step preconditioner. However, the larger reduction in the number of iterations compensates for the extra cost in each iteration.

In Table 5.4, we present a detailed performance comparison between many different techniques. (Note that the CPU clock is only accurate to within 5%.) It is clear that a careful implementation of the preconditioned Krylov space method is very important for performance. Although some extra memory is needed for the small system (5.6) (see the comparison in Table 5.3), the total memory requirement for the two-step preconditioner is still competitive with a direct method.

6. Conclusion. We emphasize that these conclusions are based on many different test cases, with widely varying bias voltages. In the course of our tests, we have solved several thousand device simulation Jacobians. Consequently, we are concerned with the best overall performance, not just isolated cases.

In agreement with previous work, we have found Bi-CGSTAB acceleration to be generally superior to either CGS or GMRES methods for device simulation Jacobians.

A generally robust iterative method for a wide range of tests used

- block scaling of the Jacobian and right-hand side,

- a block RCM ordering of the unknowns,
- block ILU preconditioning (level(1) or level(2) appeared to be a good choice),
- Bi-CGSTAB acceleration.

We found that it was important to treat the Jacobian as a block system, i.e., all the unknowns associated with a node remain tightly coupled together during the ordering and preconditioning steps. A similar effect has been observed in petroleum reservoir simulation [38]. In particular, if block scaling is not used, we have found that convergence is problematic in many cases.

Use of a two-step preconditioner which uses the usual block ILU as a first stage, followed by a more accurate treatment of the electric potential terms was slightly faster (in terms of CPU time) than the block ILU factorization by itself. More importantly, the two-step preconditioner typically reduced the number of iterations required for a solve by about one-half. Consequently, if we measure robustness by the number of iterations required, the two-stage preconditioner is more robust than the block ILU factorization by itself. If the matrix is very ill conditioned, the two-step preconditioner is more reliable than the one-step technique.

Tests using a drop tolerance method based on the scalar values of the Jacobian matrix indicated that this approach was not very robust. This is likely due to the large variation in the magnitudes of the Jacobian entries typical of device simulation. In this case, it becomes difficult to define *small* in terms of fill entry size in a meaningful way. A similar effect was observed for high Reynolds number Navier–Stokes flows [13]. However, it may be that a different choice for drop tolerance criteria would produce better results.

REFERENCES

[1] J. M. C. AARDEN AND K.-E. KARLSSON, *Preconditioned CG-type methods for solving the coupled system of fundamental semiconductor equations*, BIT, 19 (1989), pp. 916–937.

[2] O. AXELSSON, *A survey of preconditioned iterative methods for linear systems of equations*, BIT, 25 (1985), pp. 166–187.

[3] R. BANK AND D. ROSE, *Some error estimates for box method*, SIAM J. Numer. Anal., 24 (1987), pp. 777–787.

[4] R. E. BANK, T. F. CHAN, W. M. COUGHRAN, AND R. K. SMITH, *The alternate block factorization procedure for systems of partial differential equations*, BIT, 29 (1989), pp. 938–954.

[5] R. E. BANK, W. M. COUGHRAN, R. S. M. A. DRISCOLL, AND W. FICHTNER, *Iterative methods in semiconductor device simulation*, Comp. Phys. Comm., 53 (1989), pp. 201–212.

[6] A. BEHIE AND P. A. FORSYTH, *Incomplete factorization methods for fully implicit simulation of enhanced oil recovery*, SIAM J. Sci. Stat. Comput., 5 (1984), pp. 543–561.

[7] G. BRUSSINO AND V. SONNAD, *A comparison of direct and preconditioned iterative techniques for sparse unsymmetric systems of linear equations*, Internat. J. Numer. Methods Engrg., 28 (1989), pp. 801–815.

[8] P. CHIN, E. F. D'AZEVEDO, P. A. FORSYTH AND W.-P. TANG, *Preconditioned conjugate gradient methods for the incompressible Navier-Stokes equations*, Internat. J. Numer. Methods Fluids, 15 (1992), pp. 273–295.

[9] E. CHOW AND W.-P. TANG, *Storage schemes for sparse matrices with variable block size*, Workshop Report, University of Waterloo, Waterloo, Ontario, Canada, April, 1993.

[10] E. CHU, A. GEORGE, J. LIU, AND E. NG, *Sparspak: Waterloo sparse matrix package: User's guide for Sparspak-a*, Workshop Report CS-84-36, University of Waterloo, Waterloo, Ontario, Canada, 1984.

[11] S. CLIFT AND P. FORSYTH, *Linear and nonlinear iterative methods for the incompressible Navier–Stokes equations*, Internat. J. Numer. Methods Fluids, 18 (1994), pp. 229–256.

[12] S. S. CLIFT AND W.-P. TANG, *Weighted graph based ordering techniques for precondditioned conjugate gradient methods*, BIT, 35 (1995), pp. 30–47.

[13] E. F. D'AZEVEDO, P. A. FORSYTH, AND W.-P. TANG, *Drop tolerance preconditioning for incompressible viscous flow*, Internat. J. Comput. Math., 44 (1992), pp. 301–312.

[14] ———, *Ordering methods for preconditioned conjugate gradient methods applied to unstructured grid problems*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 944–961.

[15] ———, *Towards a cost-effective high order ILU preconditioner*, BIT, 32 (1992), pp. 442–463.

[16] ———, *Towards a cost-effective ILU preconditioner with high level fill*, BIT, 32 (1992), pp. 442–463.

[17] ———, *Two variants of minimum discarded fill ordering*, in Proc. IMACS Intern. Symp. on Iterative Methods in Linear Algebra, R. Beauwens and P. de Groen, eds., North-Holland, Amsterdam, 1992, pp. 603–612.

[18] R. S. DEMBO, S. C. EISENSTAT, AND T. STEIHAUG, *Inexact Newton methods*, SIAM J. Numer. Anal., 19 (1982), pp. 400–408.

[19] H. A. V. DER VORST, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 631–644.

[20] J. K. DICKINSON AND P. A. FORSYTH, *Preconditioned Conjugate Gradient Methods for three dimensional linear elasticity*, Internat. J. Numer. Methods Engrg., 37 (1994), pp. 2211–2234.

[21] S. DOI AND A. LICHNEWSKY, *A graph-theory approach for analyzing the effects of ordering on ILU preconditioning*, Research Report 1452, INIA, Le Chesnay Cedex, France, June 1991.

[22] M. DRIESSEN AND H. A. V. DER VORST, *Bi-CGSTAB in semiconductor modelling*, Simulation of Semiconductor Devices and Processes, 4 (1991), pp. 45–54.

[23] I. S. DUFF AND G. A. MEURANT, *The effect of ordering on preconditioned conjugate gradients*, BIT, 29 (1989), pp. 635–657.

[24] L. C. DUTTO, *The effect of ordering on preconditioned GMRES algorithm, for solving the compressible Navier–Stokes equations*, Internat. J. Numer. Methods Engrg., 36 (1994), pp. 457–497.

[25] C. FITZSIMONS, *An efficient implementation of Bi-CGSTAB applied to three-dimensional semiconductor device problems*, Research Report, Rutherford Appleton Laboratory, Mathematical Software Group, Chilton, Didcot, Oxfordshire, 1992.

[26] R. W. FREUND, *A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems*, SIAM J. Sci. Comput., 14 (1993), pp. 470–482.

[27] A. GEORGE AND J. W. H. LIU, *Computer solution of large sparse positive-definite systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[28] O. HEINREICHSBERGER, S. SELBERHERR, M. STIFTINGER, AND K. P. TRAAR, *Fast iterative solution of carrier continuity equations for three-dimensional device simulation*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 289–306.

[29] G. HEISER, C. POMMERELL, J. WEIS, AND W. FICHTNER, *Three-dimensional numerical semiconductor device simulation: Algorithms, architectures, results*, IEEE Trans. Computer-Aided Design Integrated Circuits Systems, 10 (1991), pp. 1218–1230.

[30] A. JENNINGS AND G. M. MALIK, *Partial elimination*, J. Inst. Maths. Appl., 20 (1997), pp. 307–316.

[31] J. R. F. MCMACKEN AND S. G. CHAMBERLAIN, *CHORD: A modular semiconductor device simulation development tool incorpolating external network models*, IEEE Trans. Computer-Aided Design, 8 (1989), pp. 826–836.

[32] T. NISHIDA AND C.-T. SAH, *A physically based mobility model for MOSFET numerical simulation*, IEEE Trans. ED, ED-34 (1987), pp. 310–320.

[33] Y. NOTAY, *Ordering methods for approximate factorization preconditioning*, Technical Report, Service de Métrologie Nucléaire, Université Libre de Bruxelles, Belgium, January 1993.

[34] C. POMMERELL AND W. FICHTNER, *New developments in iterative methods for device simulation*, Simulation of Semiconductor Devices and Processes, 4 (1991), pp. 243–248.

[35] C. S. RAFFERTY, M. R. PINTO, AND R. W. DUTTON, *Iterative methods in semiconductor device simulation*, IEEE Trans. Computer-Aided Design Integrated Circuits Systems, 4 (1985), pp. 462–471.

[36] Y. SAAD AND M. H. SCHULTZ, *GMRES: A generalized minimum residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.

[37] D. L. SCHARFETTER AND H. K. GUMMEL, *Large signal analysis of a silicon read diode oscillator*, IEEE Trans. ED, ED-16 (1969), pp. 64–77.

[38] H. D. SIMON, *Incomplete LU preconditioners for conjugate-gradient-type iterative methods*, Soc. Pet. Eng. J. Res. Eng., 3 (1988), pp. 302–306.

[39] P. SONNEVELD, *CGS, a fast Lanczos-type solver for nonsymmetric systems*, SIAM J. Sci. Stat. Comput., 10 (1989), pp. 36–52.

[40] Z.-Y. WANG, K.-C. WU, AND R. W. DUTTON, *An approach to construct pre-conditioning matrices for block iteration of linear equations*, IEEE Trans. Computer-Aided Design Integrated Circuits Systems, 11 (1992), pp. 1334–1343.

[41] K.-C. WU, R. F. LUCAS, Z.-Y. WANG, AND R. W. DUTTON, *New approaches in a 3-d one-carrier device solver*, IEEE Trans. Computer-Aided Design Integrated Circuits Systems, 8 (1989), pp. 528–537.

[42] Z.-Y. ZHAO, Q.-M. ZHANG, G.-L. TAN, AND J. M. XU, *A new preconditioner for CGS iteration in solving large nonsymmetric linear equations in semiconductor device simulation*, IEEE Trans. Computer-Aided Design Integrated Circuits Systems, 10 (1991), pp. 1432–1440.

SIAM J. SCI. COMPUT.
Vol. 17, No. 1, pp. 118–132, January 1996

A MULTIGRID PRECONDITIONER FOR THE SEMICONDUCTOR EQUATIONS*

JUAN C. MEZA† AND RAY S. TUMINARO‡

Abstract. A multigrid preconditioned conjugate gradient algorithm is introduced into a semiconductor device modeling code DANCIR. This code simulates a wide variety of semiconductor devices by numerically solving the drift-diffusion equations. The most time-consuming aspect of the simulation is the solution of three linear systems within each iteration of the Gummel method. The original version of DANCIR uses a conjugate gradient iteration preconditioned by an incomplete Cholesky factorization. In this paper, we consider the replacement of the Cholesky preconditioner by a multigrid preconditioner. To adapt the multigrid method to the drift-diffusion equations, interpolation, projection, and coarse grid discretization operators need to be developed. These operators must take into account a number of physical aspects that are present in typical devices: wide-scale variation in the partial differential equation (PDE) coefficients, small-scale phenomena such as contact points, and an oxide layer. Additionally, suitable relaxation procedures must be designed that give good smoothing numbers in the presence of anisotropic behavior. The resulting method is compared with the Cholesky preconditioner on a variety of devices in terms of iterations, storage, and run time.

Key words. multigrid, semiconductors, drift-diffusion

AMS subject classifications. 35Q35, 65F10, 65F50

1. Introduction.

Currently most integrated circuits are designed in a "trial and error" fashion. That is, prototypes are built and improved via experimentation and testing. In the near future it may be possible to significantly reduce the cost of building new devices by using computer simulations to shorten the design cycle. To accurately perform these complex simulations in three dimensions, however, new algorithms and high performance computers are necessary.

In this paper we discuss the use of multigrid preconditioning in conjunction with a conjugate gradient algorithm inside a semiconductor device modeling code DANCIR [7]. DANCIR is a three-dimensional semiconductor device simulator capable of computing the solution of the steady-state, drift-diffusion equations. The solution of the drift-diffusion equations involves the solution of a large nonlinear set of equations that arise from the spatial discretization of the drift-diffusion equations on a rectangular grid. These nonlinear equations are solved using Gummel's method, which requires three symmetric linear systems to be solved within each Gummel iteration. It is the solution of these linear systems that comprises the dominant computational cost of a simulation. The original version of DANCIR uses an incomplete Cholesky preconditioned conjugate gradient algorithm to solve these linear systems. Unfortunately, this algorithm has a number of disadvantages: (1) it can take many iterations to converge or it may not converge at all in some cases, (2) it can require a significant amount of computing time, and (3) it is not very parallelizable.

In this study we consider an alternate solution method based on a multigrid preconditioner. The multigrid method uses iterations on a hierarchy of grids to accelerate the convergence on the finest grid. The method requires interpolation, projection, and discretization operators for the different grids to be defined. Developing these operators in the context of the drift-diffusion equations requires some care due to the presence of greatly varying physical phenomena including wide-scale variation in PDE coefficients and small-scale phenomena such as contact

*Received by the editors June 20, 1994; accepted for publication (in revised form) January 30, 1995. This work was supported by the Applied Mathematical Science Program, U.S. Department of Energy, Office of Energy Research and was performed at Sandia National Laboratories operated for the U.S. Department of Energy under contract DE-AC04-94AL85000.

†MS 9214, Sandia National Laboratories, P.O. Box 969, Livermore, CA 94551 (meza@ca.sandia.gov).

‡MS 1110, Sandia National Laboratories, P.O. Box 5800, Albuquerque, NM 87185 (tuminaro@cs.sandia.gov).

points. In both cases, the development of operator-dependent interpolation and projection is essential for improving the performance. Further, the presence of certain device characteristics such as oxide layers requires some care to ensure that the different operators and the grid hierarchy adequately approximate the PDE on all levels. Finally, the presence of a severely stretched grid gives rise to anisotropic phenomena that requires a suitable relaxation procedure.

The paper is organized as follows. Section 2 describes the drift-diffusion equations that are most commonly used to model semiconductor devices. We also present the numerical methods currently used in a particular simulation code developed at Sandia National Laboratories. In §3 we describe the multigrid preconditioner as well as some of the choices necessary to implement this algorithm for the drift-diffusion equations. In §4, the resulting method is compared with the incomplete Cholesky preconditioner on a variety of devices in terms of iterations, storage, and run time. We conclude in §5 with a summary of our results.

2. The drift-diffusion equations. The drift-diffusion model for semiconductor device modeling consists of a set of three coupled, nonlinear PDEs: the potential equation plus two continuity equations, one each for the electron and hole current densities. For a complete derivation of the equations the reader can consult a variety of references, for example, [9], [11], [13].

The potential or Poisson equation is given by

$$\epsilon \nabla \cdot E = -\epsilon \nabla^2 \psi = \rho, \tag{2.1}$$

where ϵ is the scalar permittivity of the semiconductor, ψ is the electric potential, and $E = -\nabla \psi$ is the electric field. The total electric charge density ρ is given by

$$\rho = q(p - n + N_D - N_A), \tag{2.2}$$

where q is the elementary charge, n is the density of free electrons, p is the density of holes, N_D is the density of donor impurities, and N_A is the density of acceptors.

The continuity equations for the electron and hole currents can be stated as

$$\frac{\partial n}{\partial t} - \nabla \cdot (\mu_n n E + D_n \nabla n) - R = 0, \tag{2.3}$$

$$\frac{\partial p}{\partial t} + \nabla \cdot (\mu_p p E - D_p \nabla p) - R = 0, \tag{2.4}$$

where μ_n and μ_p are the electron and hole mobilities, respectively, D_n and D_p are the diffusion coefficients, and R is a term that accounts for the recombination and generation of electrons and holes. The mobilities and the recombination-generation term are functions of various physical and semi-empirical parameters and are usually modeled through a variety of sophisticated methods (see, for example, [11]).

2.1. Scaling of variables. The wide range in magnitude of both the dependent and independent variables creates difficulties in the numerical solution process. Independent variables such as the concentrations of impurity dopings N_D and N_A range from 10^{13} to 10^{19} carriers per cubic centimeter. Dependent variables such as the carrier densities n and p can range from 10^3 to 10^{19} carriers per cubic centimeter. The difficulties associated with the wide range in the magnitude of the dependent variables can be circumvented to a certain extent by employing different variables. However it has been noted by Polak [12] that changing variables amounts to trading high variability in the dependent variables for increased nonlinearity in the equations.

In the DANCIR code, the carrier concentrations are scaled by using the Slotboom variables u and v:

$$n = n_{ie} \exp\left(\frac{q\psi}{kT}\right) u, \tag{2.5}$$

$$p = n_{ie} \exp\left(\frac{-q\psi}{kT}\right) v, \tag{2.6}$$

where n_{ie} is the effective intrinsic carrier concentration, k is the Boltzmann constant, and T is the bulk material temperature.

One of the advantages of using the Slotboom variables is that the current continuity equations assume the form of Poisson equations facilitating the numerical solution process. Using the Slotboom scaling the current densities can be written as

$$J_n = kT\mu_n n_{ie} \exp(q\psi/kT)\nabla u, \tag{2.7}$$

$$J_p = -kT\mu_p n_{ie} \exp(-q\psi/kT)\nabla v. \tag{2.8}$$

The resulting steady-state, drift-diffusion equations can then be expressed as

$$f_1(\psi, u, v) = \epsilon\nabla^2\psi + q\left[n_{ie}e^{\frac{-q\psi}{kT}} v - n_{ie}e^{\frac{q\psi}{kT}} u + N_D - N_A\right] = 0, \tag{2.9}$$

$$f_2(\psi, u, v) = \nabla\cdot\left[kT\mu_n n_{ie}e^{\frac{q\psi}{kT}}\nabla u\right] + R = 0, \tag{2.10}$$

$$f_3(\psi, u, v) = \nabla\cdot\left[kT\mu_p n_{ie}e^{\frac{-q\psi}{kT}}\nabla v\right] + R = 0. \tag{2.11}$$

2.2. Nonlinear equations. If the functions f_1, f_2, and f_3 are defined by equations (2.9)–(2.11) then the nonlinear system of equations arising from the spatial discretization of the drift-diffusion equations can be written more compactly as $F(\psi, u, v) = [f_1, f_2, f_3]^T = 0$. The DANCIR code uses Gummel's method [6] (also known as nonlinear block Gauss–Seidel) to solve this nonlinear equation. One Gummel iteration consists of solving f_2 for u, f_3 for v, and then f_1 for ψ. Gummel's method has the advantage of only having to solve three linear systems at each iteration. The disadvantage is that convergence can be quite slow in certain circumstances, for example in high voltage situations.

In practice, the solution of the steady-state, drift-diffusion equations is accomplished by solving a series of continuation steps where each continuation step is in turn a steady-state problem. The initial steady-state or equilibrium problem solved is that of the device with no external voltages applied. The potential at the contacts is then incremented until the desired voltage is reached at the contacts. The initial estimate for the potential is computed by solving a nonlinear potential equation. The DANCIR code uses a Newton method for this calculation because the Jacobian is symmetric in this special case, thereby not incurring any extra expense for storage over the Gummel iteration.

2.3. Linear equations. Within each Gummel iteration, three linear systems of equations must be solved. As we mentioned above, the use of the Slotboom variables transforms the continuity equations for the electron and hole currents into a set of self-adjoint PDEs. The linear systems resulting from the discretization are therefore symmetric and positive definite. The particular method used in the original version of the DANCIR code is a preconditioned

```
Procedure MG($b$, $u$, $level$)
    if ($level$ == COARSEST ) then $u \leftarrow A_{level}^{-1} b$
    else
        $u \leftarrow$relax($b$, $u$, $level$)
        $r \leftarrow b - Au$
        $\tilde{r} \leftarrow Rr$          /* R is a projection operator */
        $v \leftarrow 0$
        $v \leftarrow$ MG($\tilde{r}$, $v$, $level + 1$)
        $u \leftarrow u + Pv$   /* P is an interpolation operator */
        $u \leftarrow$relax2($b$, $u$, $level$)
    endif
```

FIG. 1. *Multigrid algorithm for* $A_{\text{level}}u = b$.

conjugate gradient method with an incomplete Cholesky factorization used as the preconditioner [4]. The solution of these linear systems usually constitutes the dominant amount of work, so any improvement in this part of the code would have a significant effect on the overall performance.

In the remainder of this paper, we discuss the replacement of this preconditioner by a multigrid preconditioner and make comparisons between the two preconditioners.

3. Multigrid preconditioner. The multigrid algorithm is a fast and efficient method for solving the systems of equations that arise from many PDE applications. We give only a brief sketch of one type of multigrid algorithm. Detailed descriptions of more general multigrid algorithms can be found in [3], [5].

One iteration of a simple multigrid "V" cycle consists of smoothing the error using a relaxation technique (such as Gauss–Seidel), "solving" an approximation to the smooth error equation on a coarse grid, interpolating the error correction to the fine grid, and finally adding the error correction into the approximation (and perhaps performing some additional relaxation steps). An important aspect of the multigrid method is that the coarse grid solution can be approximated by recursively using the multigrid idea. That is, on the coarse grid, relaxation is performed to reduce high frequency errors followed by the projection of a correction equation on yet a coarser grid, and so on. Thus, the multigrid method corresponds to a series of relaxation iterations on a hierarchy of grids with different mesh sizes followed by the use of a direct solver on the coarsest grid (which is usually a fairly small grid). We summarize one iteration of this procedure in Fig. 1.

It is important to note that when the multigrid method is used as a preconditioner within the conjugate gradient method, it is necessary that the preconditioner be symmetric. This can be accomplished in the above procedure by choosing the postrelaxation ("relax2" in Fig. 1) as the transpose of the prerelaxation ("relax" in Fig. 1) [8]. The Jacobi iteration is one smoother that has this property when it is used for prerelaxation and postrelaxation. Another smoother combination with this property consists of using red-black Gauss–Seidel for prerelaxation and black-red Gauss–Seidel for postrelaxation.

To complete the definition of the above method, we will define a grid hierarchy, grid transfer operators, a coarse grid discretization scheme, and a specific relaxation method.

3.1. Grid hierarchy. In the DANCIR code, the discretization mesh used consists of a tensor product grid of one-dimensional arrays. In defining a grid hierarchy for the multigrid method, we consider only coarsening by a factor of 2 to facilitate code development. It should be noted that in order to carry out this procedure it is necessary to restrict the size of the fine grid.

Specifically, for two-dimensional simulations, the fine grid must contain $(2^k m+1) \times (2^j n+1)$ points where $\max(k, j)$ defines the number of levels in the hierarchy and $(m+1) \times (n+1)$ is the size of the coarsest grid. For the case $k \geq j$, the grid hierarchy is defined as follows:

$$\begin{aligned}
\mathcal{G}_0 &: 2^k m+1 \times 2^j n+1, \\
\mathcal{G}_1 &: 2^{(k-1)} m+1 \times 2^{(j-1)} n+1, \\
&\vdots \\
\mathcal{G}_j &: 2^{(k-j)} m+1 \times n+1, \\
&\vdots \\
\mathcal{G}_k &: m+1 \times n+1.
\end{aligned}$$

Though this scheme is straightforward, some care must be taken due to certain device characteristics such as the presence of an oxide layer as well as the contacts. This difficulty will be discussed after defining the grid transfer operators and the coarse grid discretization.

3.2. Grid transfer operators. In simple multigrid codes it is quite common to use linear interpolation for second-order problems and full weighting or half weighting for projection [3]. In most cases these simple grid transfer operators are sufficient and the resulting multigrid scheme works quite well. However, when the PDE coefficients vary greatly (or contain a discontinuity), these choices for the grid transfer operators may not be sufficient. To illustrate this point consider the simple PDE

$$(w(x)\, u_x\,)_x = f(x), \qquad 0 < x < 1, \tag{3.1}$$

with Dirichlet boundary conditions at $x = 0$ and $x = 1$ and

$$w(x) = \begin{cases} \varepsilon & x < .5, \\ 1 & x \geq .5. \end{cases}$$

If linear interpolation (or even higher-order interpolation) is used to obtain the value of u at $x = .5$, the resulting quantity $w(x)u_x$ will in general be discontinuous. Since this term is differentiated, it is quite clear that this discontinuity is undesirable. That is, interpolating such that u is smooth (i.e., u_x is constant) at the interpolation point is not the right criterion. Instead, we need to take into account the function $w(x)$. One possibility (see [14] or [1]) is to require that the term $w(x)u_x$ be constant at the interpolated points. This results in an interpolation formula of the form

$$u(x) = \frac{w(x-\frac{h}{2})u(x-h) + w(x+\frac{h}{2})u(x+h)}{w(x+\frac{h}{2}) + w(x-\frac{h}{2})} \tag{3.2}$$

for the above example on a uniform grid (with mesh spacing h on the fine grid). This formula can be verified by observing that when it is used, the central difference approximations to $w(x)u_x$ at $x - \frac{h}{2}$ and $x + \frac{h}{2}$ are equal. It should be noted that the above interpolation operator could have been defined in simply algebraic terms. Specifically, when (3.1) is discretized in a standard way the resulting difference operator is tridiagonal. To solve this tridiagonal system, a procedure called cyclic reduction can be used [4]. This procedure is essentially Gaussian elimination with a special elimination ordering of the points. The first step corresponds to eliminating all the even numbered points (where the points are numbered sequentially along the line). If we make an analogy between the multigrid algorithm and cyclic reduction, the interpolation algorithm corresponds to one step in the back solve of cyclic reduction.

Generalization of the interpolation procedure given above to two- or three-dimensional PDEs is not obvious because the exact analogy with cyclic reduction is no longer possible.

In this work we consider the following interpolation procedure in two dimensions.

1. Split the finite difference operator on $\mathcal{G}_l$, A_l, into matrices corresponding to the x and y derivatives as well as the 0th order terms:

$$A_l = A_l^x + A_l^y + G_l.$$

2. Average the derivative operators in the direction orthogonal to the derivative

$$\tilde{A}_l^x(j) = \frac{1}{4}A_l^x(j-1) + \frac{1}{2}A_l^x(j) + \frac{1}{4}A_l^x(j+1),$$

where $A_l^x(j)$ is the x derivative operator for level l on horizontal line j of the grid.

3. Interpolate in the x direction. For points where the coarse and fine grid coincide, use injection. For points in between two coarse grid points use the one-dimensional procedure outlined above in conjunction with $\tilde{A}_l^x(i)$. Specifically, set the interpolated value at (i, j) such that

$$\{\tilde{A}_l^x(j)u\}_i = 0,$$

where u is the vector of interpolated values corresponding to line j of the grid. This essentially corresponds to the back solve step of cyclic reduction applied to $\tilde{A}_l^x(j)$. It is important to note that at this point the operator u is defined on every other horizontal line.

4. Repeat the procedure in the y direction for the horizontal lines that have not yet been defined in the previous x interpolation. In particular, for fine grid points that are surrounded by four coarse grid points, use the interpolated values from the x interpolation.

Effectively, this procedure corresponds to performing the interpolation in one direction after another using one-dimensional difference operators defined from the original difference operator. The overall procedure is similar to that in [1], where they use an arithmetic average of the harmonic averages (as opposed to the harmonic average of the arithmetic averages) to define the interpolation operator.

For the projection operator we simply use the transpose of the interpolation scheme. Given that the difference operator is symmetric, this is a quite natural way to define a weighted projection operator. Using this projection on a one-dimensional problem corresponds to performing the forward elimination step of a cyclic reduction procedure on the tridiagonal discretization matrix.

3.3. Coarse grid discretizations. In typical multigrid codes, there are two ways of obtaining PDE discretizations for the coarser grids. One reliable technique, Galerkin coarsening, uses the interpolation and projection operators.

While this procedure works well for two- and three-dimensional problems, it has several disadvantages. For example, when standard interpolation and projection operators are used the Galerkin procedure may result in a larger difference stencil on coarser grids. Thus the programmer would have to write new codes for the coarse grid discretization as well as incur an extra storage penalty.

An alternative to the Galerkin procedure is to use an averaging procedure to generate PDE coefficients in conjunction with the differencing scheme used on the fine grid. There are many possible averaging procedures. In this work, we use a scheme based on the one-dimensional Galerkin operator. Specifically, when a Galerkin procedure is used in conjunction with the operator-dependent interpolation and projection on (3.1), the resulting coarse grid operator is equivalent to the reduced operator obtained with one step of cyclic reduction. For (3.1), this

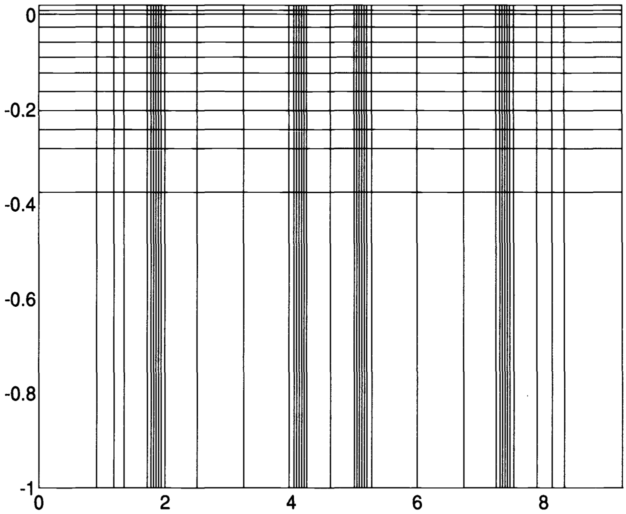


FIG. 2. *Grid obtained by keeping every eighth line from a typical MOSFET grid.*

coarse grid operator is equivalent to a three-point difference operator that corresponds to using the fine grid difference scheme on the coarse grid in conjunction with the PDE coefficients

$$\tilde{w}(x) = \frac{2\ w(x+h)\ w(x-h)}{w(x+h) + w(x-h)}.$$

For higher-dimensional problems we first split the finite difference operator into different terms. A standard averaging procedure is used to approximate the constant term G_l on the coarse grid. For the derivative terms, we first form averages (e.g., $\tilde{A}_l^x(i)$) as for the interpolation. Then, we use a cyclic reduction procedure on the one-dimensional problems in each of the different coordinate directions to generate approximations to the PDE coefficients on the coarser grid. Overall, the resulting procedure can be viewed as an approximation to the reduced equations of cyclic reduction (or the Galerkin coarsening) where the operator has first been split into component terms, the cyclic reduction procedure has been applied, and then the terms are regrouped to form the coarse grid discretization.

3.4. Relaxation method. There are two major possibilities for the relaxation method: point relaxation and line relaxation. It is well known in the multigrid community that line relaxation (or semicoarsening) should be adopted when the method is applied to anisotropic problems. That is, while standard point relaxation methods sufficiently smooth the error for the Poisson equation, their performance is quite poor for severely anisotropic problems. If, however, line relaxation is used instead of the point relaxation, it is once again possible to obtain good multigrid convergence rates even for anisotropic problems. We omit the details and refer the reader to [3], where a Fourier analysis is given illustrating this phenomenon. In our case, the highly stretched grid (e.g., Fig. 2) is a source of anisotropic phenomena.[1] Thus, we have implemented red-black Gauss–Seidel as well as red-black alternating line Gauss–Seidel for prerelaxation and black-red point and line Gauss–Seidel for postrelaxation. Additionally, we have implemented local Gauss–Seidel procedures to take into account the physics of the simulation. In particular, the majority of the difficulties for the smoother are caused by the

[1] It is not clear whether line relaxation would be necessary with another grid structure. That is, if the sole source of the anisotropic behavior is the grid, it may not be necessary to use line relaxation.

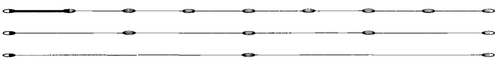


FIG. 3. *One-dimensional grid hierarchy.*

top of the device (near the contacts, oxide layer, and doped regions). The local Gauss–Seidel procedures consist of performing red-black (point or alternating line) Gauss–Seidel on the residual equation restricted to the domain covering the upper one-fourth of the device. This local procedure is usually used after the standard relaxation procedure to improve the smoother in the upper part of the device (at only one-fourth the cost of the global procedure).

3.5. Contact points and oxide layer. In defining the grid hierarchy no particular attention has been given to device characteristics such as the oxide layer and the contact points present in many devices. It is entirely possible that the location of the boundary between the oxide and the interior or the boundary between a contact point and the interior could differ between the fine and coarse grids. For example, consider the one-dimensional hierarchy of three grids shown in Fig. 3. If we have a contact region (denoted in bold) that covers two points on the fine grid, then it effectively becomes smaller on the next coarser grid. To prevent this phenomenon, there are two possible solutions. One option is to choose the grid hierarchy in conjunction with the contacts (and oxide) such that the boundaries of these regions remain in place. The other possibility, which we consider, is to keep the grid hierarchy but to modify the discretization and the right-hand side on the coarse grid such that the location of the contact boundary is maintained. For example, the matrix equation corresponding to the fine grid above

$$\begin{pmatrix} 1 &&&&&&&& \\ & 1 &&&&&&& \\ && -2 & 1 &&&&& \\ && 1 & -2 & 1 &&&& \\ &&& 1 & -2 & 1 &&& \\ &&&& 1 & -2 & 1 && \\ &&&&& 1 & -2 & 1 & \\ &&&&&& 1 & -2 & \\ &&&&&&&& 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \\ b_8 \\ b_9 \end{pmatrix}$$

has the same solution at the odd points as

$$\begin{pmatrix} 1 &&&& \\ & \mathbf{-3} & \mathbf{1} && \\ & 1 & -2 & 1 & \\ && 1 & -2 & \\ &&&& 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_3 \\ u_5 \\ u_7 \\ u_9 \end{pmatrix} = \begin{pmatrix} b_1 \\ \mathbf{2b_3 + b_4} \\ b_4 + 2b_5 + b_6 \\ b_6 + 2b_7 + b_8 \\ b_9 \end{pmatrix}.$$

Notice that the main change is the second equation (discretization and right-hand side projection). This procedure corresponds to cyclic reduction and can also be motivated by truncation error arguments. The main point is that the operator-dependent schemes automatically take care of these situations to maintain a proper coarse grid approximation. Of course, this example is one dimensional and in higher dimensions it will not be possible to have an exact representation on the coarse grid. However, by using the operator-dependent grid transfers and discretizations we need not worry too much about the grid hierarchy with respect to the oxide and the contacts. Further, any small degradations in the coarse discretization (for example,

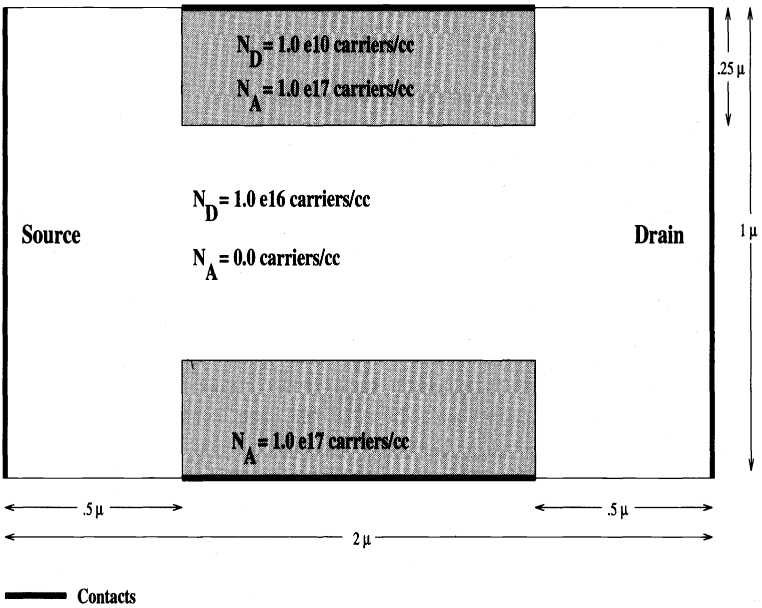

N_D = 1.0 e10 carriers/cc
N_A = 1.0 e17 carriers/cc
N_D = 1.0 e16 carriers/cc
N_A = 0.0 carriers/cc
Source
Drain
N_A = 1.0 e17 carriers/cc
.25 μ
1 μ
.5 μ
.5 μ
2 μ
Contacts

FIG. 4. *JFET device.*

near the corners of an oxide region) will in general not cause great difficulties for a conjugate gradient routine (because the problem areas will be of very low rank).

4. Algorithm summary. To summarize, the numerical algorithm used for the nonlinear equations is the Gummel iteration. Gummel's method requires three linear system solutions. For these a conjugate gradient algorithm is used in conjunction with a multigrid preconditioner. Finally, within the multigrid preconditioner, Gauss–Seidel is used as a smoother with red-black sweeps for prerelaxation and black-red for postrelaxation. It should be noted that it is possible to use multigrid as a solver instead of a preconditioner (i.e., without the conjugate gradient iterations). However, in our experiments we found that the multigrid preconditioned conjugate gradient code outperformed the use of just multigrid. We believe that there are two primary reasons for this:

- The presence of small-scale effects such as small contacts and doping regions that disappear on the coarse grid degrade the multigrid coarse grid operator.
- The interpolation, projection, and coarse grid scheme used are a bit simplistic. These choices were made partially because they were easy to incorporate into a large already-existing program with fairly complicated data structures and a simplistic scheme for handling nonrectangular grids (resulting from the oxide region). Unfortunately these procedures do not handle the corners of the oxide very well.

While these low-rank degradations can make the multigrid convergence slower, they do not greatly affect the conjugate gradient procedure.

5. Numerical results. A two-dimensional version of the multigrid algorithm described in the previous section was incorporated into the DANCIR code. To evaluate the multigrid scheme, a number of experiments have been performed on a variety of devices. Before illustrating these results we briefly describe the devices used in the comparison.

5.1. Junction field effect transistor (JFET). The first model used consists of a JFET depicted in Fig. 4. The simulation is started with zero volts on all of the contacts. Once

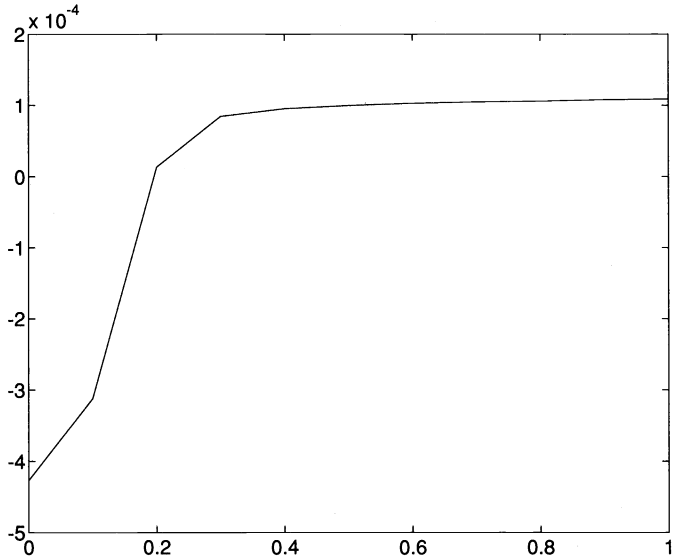


FIG. 5. *Current–voltage characteristics for JFET.*

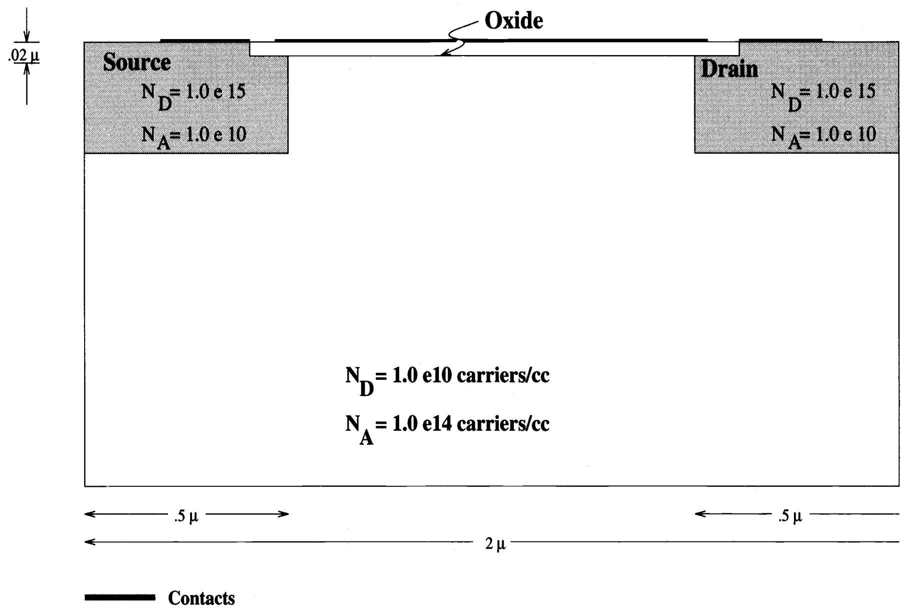


FIG. 6. *MOSFET1 device.*

the equilibrium solution is computed, the voltage on the bottom and top contacts is gradually incremented (and the corresponding steady-state solutions are computed) until the final contact voltage is reached (1.0 volts). After this stage the drain contact voltage is incremented until it attains a value of 1.0 volts. The current–voltage characteristics (computed with the DANCIR/MG code) are displayed in Fig. 5.

5.2. MOSFET1. The second test problem consists of a simple MOSFET device depicted in Fig. 6. In this case the simulation starts at equilibrium. The gate contact is first incremented to 5.5 volts and finally the drain value is incremented to 2.5 volts.

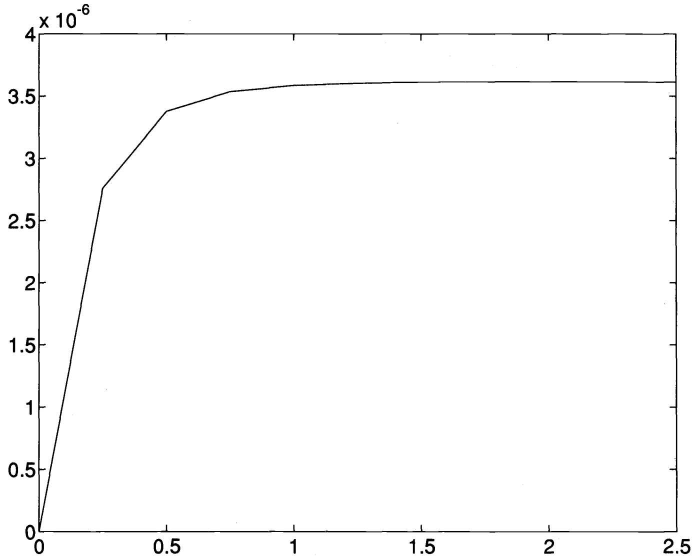


FIG. 7. *Current–voltage characteristics for MOSFET1.*

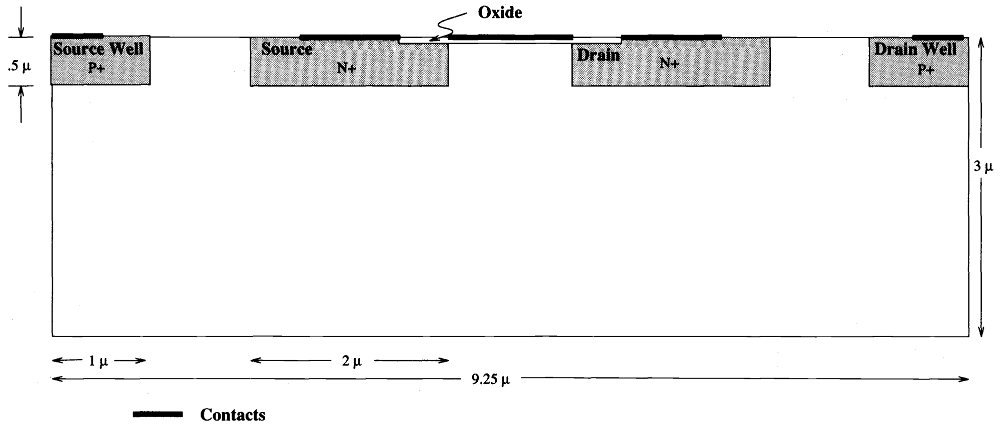


FIG. 8. *MOSFET2 device.*

The current–voltage characteristics (computed with the DANCIR/MG code) are displayed in Fig. 7.

5.3. MOSFET2. This final device is our most complex device and corresponds to a 1.25μ N-channel MOSFET that has been used at Sandia for testing purposes. This device is depicted in Fig. 8. In this case the simulation starts at equilibrium. The gate contact is first incremented to 5.5 volts and finally the drain value is incremented to 2.5 volts.

The current–voltage characteristics (computed with the DANCIR/MG code) are displayed in Fig. 9. These characteristics are standard for this device.

In Tables 1–3 we illustrate the total number of iterations for the conjugate gradient method and the CPU time corresponding to an entire simulation where the conjugate gradient iterations for each linear solve terminate when the residual is reduced by 10^{-9}. In parenthesis we indicate the average number of conjugate gradient iterations per linear solve. It should be noted that a less strict convergence criterion for the linear subiterations is not considered in this paper.

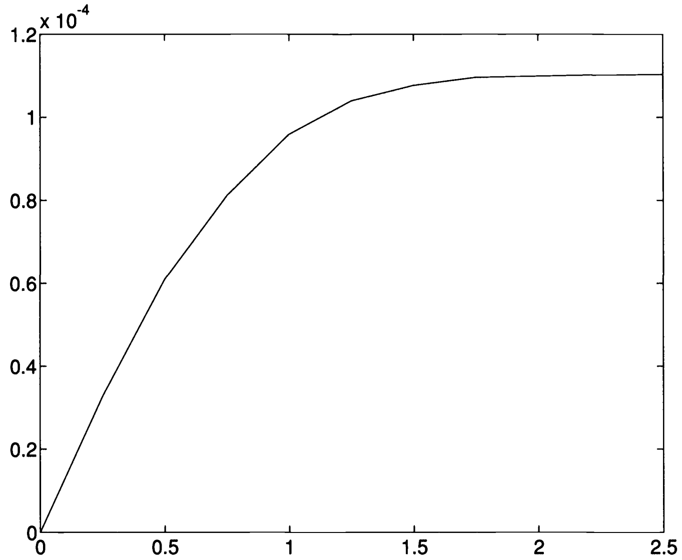


FIG. 9. *Current–voltage characteristics for MOSFET2.*

TABLE 1
JFET results: iterations and time (in minutes) for 23 Gummel iterations.

	49 × 33				321 × 97			
Preconditioner	Poisson		Continuity		Poisson		Continuity	
	its	time	its	time	its	time	its	time
ILU	1499 (13)	.4	11351 (48)	2.9	5243 (42)	25.0	49635 (199)	231.1
MG(1L,0L,4)	726 (6)	0.7	1959 (8)	1.9	1270 (10)	29.2	2182 (9)	50.4
MG(1L,1L,4)	562 (5)	0.7	1750 (7)	2.3	1008 (8)	30.3	1750 (7)	52.9
MG(2L,2L,4)	437 (4)	1.0	1499 (6)	3.3	840 (7)	41.0	1502 (6)	73.4
MG(1L,3L,4)	456 (4)	0.9	1589 (7)	3.0	815 (7)	36.0	1488 (6)	65.9
MG(3P,0P,4)	963 (8)	0.8	4133 (18)	3.3	1192 (10)	23.5	1827 (7)	36.5
MG(5P,5P,4)	626 (5)	1.0	2650 (11)	4.1	973 (8)	34.6	1555 (6)	55.9
MG(1L,2L,4)	503 (4)	0.8	1682 (7)	2.7	887 (7)	33.0	1543 (6)	57.6
MG(1L,2L,5)	503 (4)	0.8	1682 (7)	2.7	887 (7)	31.5	1550 (6)	55.1
MG(1L,2L,6)	503 (4)	0.8	1682 (7)	2.7	887 (7)	31.3	1606 (6)	56.7
MG(1L,0L,5)	726 (6)	0.6	1959 (8)	1.7	1273 (10)	23.7	2191 (9)	40.9
MG(1L,0L,6)	726 (6)	0.6	1959 (8)	1.7	1273 (10)	23.7	2214 (9)	41.3

TABLE 2
MOSFET1 results: iterations and time (in minutes) for 20 Gummel iterations.

	129 × 129			
Preconditioner	Poisson		Continuity	
	its	time	its	time
ILU	21703 (139)	53.9	61504 (197)	152.6
MG(1L,0L,4)	1635 (10)	17.4	3755 (12)	39.7
MG(1L,1L,4)	1523 (10)	20.8	3369 (11)	45.9
MG(2L,2L,4)	1485 (10)	33.4	2876 (9)	64.5
MG(1L,3L,4)	1440 (9)	28.5	3165 (10)	62.4
MG(5P,5P,4)	No convergence			
MG(1L,2L,4)	1494 (10)	25.0	3261 (10)	54.4
MG(1L,2L,5)	1511 (10)	25.0	3292 (11)	54.5
MG(1L,2L,6)	1510 (10)	25.0	3311 (11)	54.8
MG(1L,0L,5)	1691 (11)	15.5	3827 (12)	35.0
MG(1L,0L,6)	1704 (11)	15.5	3883 (12)	35.2

TABLE 3
MOSFET2 results: iterations and time (in minutes) for 352 Gummel iterations.

Preconditioner	321 × 97			
	Poisson		Continuity	
	its	time	its	time
ILU	21339 (61)	97.0	171015 (243)	770.2
MG(1L,0L,4)	3202 (9)	69.3	13774 (20)	287.9
MG(1L,1L,4)	2643 (8)	71.7	11212 (16)	295.5
MG(2L,2L,4)	2231 (6)	95.1	9336 (13)	389.3
MG(1L,3L,4)	2204 (6)	84.2	10569 (15)	392.6
MG(5P,5P,4)	No convergence			
MG(1L,2L,4)	2446 (7)	79.0	10352 (15)	325.6
MG(1L,2L,5)	2446 (7)	74.0	11961 (17)	359.0
MG(1L,2L,6)	2446 (7)	74.0	11024 (16)	331.8
MG(1L,0L,5)	3208 (9)	59.8	15593 (22)	289.0
MG(1L,0L,6)	3208 (9)	59.8	14390 (20)	267.0

TABLE 4
Total simulation time.

	JFET		MOSFET1	MOSFET2
Preconditioner	49 × 33	321 × 97	129 × 129	321 × 97
ILU	5.1	321.7	231.5	945.8
MG(1L,0L,4)	4.3	114.4	75.1	424.0
MG(1L,0L,5)	4.3	99.5	68.3	406.5
MG(1L,0L,6)	4.3	100.1	68.5	391.1
MG(1L,1L,4)	4.8	142.2	85.4	432.5
MG(2L,2L,4)	6.1	153.1	118.0	555.2
MG(1L,3L,4)	5.2	139.1	110.6	544.2
MG(1L,2L,4)	5.3	126.8	98.6	470.9
MG(1L,2L,5)	5.3	122.7	98.6	499.7
MG(1L,2L,6)	5.6	124.1	99.1	472.5
MG(3P,0P,4)	5.9	92.5	nc	nc
MG(5P,5P,4)	7.1	126.3	nc	nc

While a milder criteria may be more efficient for the overall nonlinear problem,[2] our focus is to compare linear solvers where it is best if both simulations (multigrid and incomplete Cholesky preconditioned) follow the same nonlinear path. In Tables 1–4 the notation MG(#1X, #2Y, #3) indicates that the multigrid preconditioner uses #1 pre- and postrelaxation iterations[3] of the "X" (P for point or L for alternating line) Gauss–Seidel procedure and #2 pre- and postrelaxation iterations of the "Y" (P for point or L for alternating line) local Gauss–Seidel procedure on each grid in the #3 level hierarchy. All of the ILU results correspond to ILU(0), which has the same sparsity pattern as the original finite difference matrix.[4] As Tables 1–4 illustrate, the multigrid code with point relaxation is ineffective for the harder problems (though it works quite well on the JFET problem). On the other hand, the multigrid with line relaxation can be a very effective solver because it requires far fewer iterations than the corresponding ILU preconditioned code and because the overall run time is much better than the ILU code even

[2] In our experience, we have found that it can often be more efficient to perform just a few multigrid sweeps for the two carrier equations. However, it is necessary to solve somewhat accurately the Poisson equation for the potential. This is due to the fact that the potential is exponentiated in the Gummel iteration. Thus, if the potential is not accurate (and these inaccuracies are amplified due to exponentiation), the nonlinear path taken by the Gummel iteration can be suboptimal.

[3] Actually, the postrelaxation operator is the transpose of the prerelaxation operator. Thus, the prerelaxation operator uses red-black Gauss–Seidel while the postrelaxation operator uses black-red Gauss–Seidel.

[4] The original DANCIR code allows for ILU(1), ILU(2), etc. Over a wide variety of numerical experiments, we have found that the total run time is usually about the same using ILU(1) as opposed to ILU(0) (though the number of iterations is less using ILU(1)).

though the cost per iteration is greater. Specifically, for the larger grid JFET problem and the smaller grid MOSFET1 and MOSFET2 problems the multigrid code is between a factor of two and four times faster than the ILU code. More importantly, the number of iterations per linear solve is relatively independent of the grid size when using the multigrid preconditioner while it grows for the ILU scheme.[5] Thus, the savings associated with multigrid are even greater for larger grids.

In terms of storage, our multigrid scheme requires slightly more storage than the ILU method. Specifically, the ILU scheme requires the storage of the ILU preconditioner ($3n$ for five-point symmetric difference operators where n is the number of grid points). In the multigrid scheme, we need additional storage for the coarse grid versions of the matrix, the solution, and the right-hand side ($\approx 3n/4$ for the coarse matrices and $\approx 2n/4$ for the coarse solutions and right-hand sides). Thus, the ILU requires an additional $3n$ values while multigrid requires an additional $5n/4$ values. However, in our multigrid implementation we also store $2(4n/3)$ intermediate values for interpolation and projection operators and $2(4n/3)$ intermediate values for the factorization of the line solver.

We conclude this section by illustrating the total CPU time for each device simulation in Table 4. By comparison with the earlier tables, the reader can verify that the conjugate gradient iteration dominates the calculation and thus the multigrid savings are significant with respect to the entire simulation time.

6. Conclusions. We have incorporated a multigrid preconditioner into a semiconductor device modeling code. To do this, a multigrid scheme was developed that could be used in cases that exhibited highly variable PDE coefficients and anisotropic behavior. In addition, special consideration had to be taken with respect to certain device characteristics such as oxide layers and small contact regions. The resulting scheme is fast, parallel, and requires far fewer iterations than the ILU scheme that it replaced. In our sample problems we improved the performance by a factor of between 2 and 4 over the ILU preconditioner. Extensions to the three-dimensional case are straightforward and planned for the future.

Finally, we note that while we have used a multigrid solver for the linear equations that arise within Gummel's method, there are potentially much greater savings if the Gummel technique can be replaced by a nonlinear multigrid iteration. We have not pursued this, but we hope that this study will give insight into this possibility.

REFERENCES

[1] R. Alcouffe, A. Brandt, J. Dendy, and J. Painter, *The multigrid methodfor diffusion equations with strongly discontinuous coefficients*, SIAM J. Sci. Statist. Comput., 2 (1981), pp. 430–454.

[2] B. Bachmann, *A Multigrid Solver for the Semiconductor Equations*, Tech. report, Institut fur Angewandte Mathematik der Universitat Zurich, Ramistr. 74, 8001 Zurich, Switzerland, 1993.

[3] A. Brandt, *Multi-level adaptive solutions to boundary-value problems*, Math. Comp., 31 (1977), pp. 333–390.

[4] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, 1983.

[5] W. Hackbusch, *Multi-grid Methods and Applications*. Springer-Verlag, Berlin, 1985.

[6] H. K. Gummel, *A self-consistent iterative scheme for one-dimensional steady state transistor calculations*, IEEE Trans. Electron Devices, ED-11 (1964), pp. 455–465.

[7] J. C. Meza and J. F. Grcar, *DANCIR: A Three-Dimensional Steady-State Semiconductor Device Simulator*, Tech. report Sand89-8266, Sandia National Laboratories, Livermore, CA, 1990.

[8] R. Kettler and J. A. Meijerink, *A Multigrid Method and a Combined Multigrid-Conjugate Gradient Method for Elliptic Problems with Strongly Discontinuous Coefficients in General Domains*, Tech. report, Shell Publ. 604, KSEPL, Rijswijk, The Netherlands, 1981.

[5] The number of multigrid iterations grows slightly for the JFET problem because the smaller grid is very easily solved by both the ILU and the multigrid schemes.

[9] M. S. Mock, *Analysis of Mathematical Models of Semiconductor Devices*, Boole Press, Dublin, 1983.

[10] J. Molenaar, *Multigrid Methods for Semiconductor Device Simulation*, Ph.D. thesis, Centrum voor Wiskunde en Informatica, Amsterdam, 1992.

[11] S. Selberherr, *Analysis and Simulation of Semiconductor Devices*, Springer-Verlag, New York, 1984.

[12] S. J. Polak, C. Den Heijer, W. H. A. Schilders, and P. Markowich, *Semiconductor device modelling from the numerical point of view*, Internat. J. Numer. Methods Engrg., 24 (1987), pp. 763–838.

[13] S. M. Sze, *Physics of Semiconductor Devices*, Wiley, New York, 1981.

[14] P. Wesseling, *An Introduction to Multigrid Methods*, Wiley, West Sussex, 1992.

SIAM J. SCI. COMPUT.
Vol. 17, No. 1, pp. 133–155, January 1996

010

MULTIGRID WAVEFORM RELAXATION ON SPATIAL FINITE ELEMENT MESHES: THE DISCRETE-TIME CASE*

JAN JANSSEN† AND STEFAN VANDEWALLE‡

Abstract. The efficiency of numerically solving time-dependent partial differential equations on parallel computers can be greatly improved by computing the solution on many time levels simultaneously. The theoretical properties of one such method, namely the discrete-time multigrid waveform relaxation method, are investigated for systems of ordinary differential equations obtained by spatial finite-element discretisation of linear parabolic initial-boundary value problems. The results are compared to the corresponding continuous-time results. The theory is illustrated for a one-dimensional and a two-dimensional model problem and checked against results obtained by numerical experiments.

Key words. parabolic partial differential equations, waveform relaxation, multigrid, linear multistep methods

AMS subject classifications. 65F10, 65L05, 65M55, 65M60

1. Introduction. We consider the numerical solution of a linear parabolic initial-boundary value problem, spatially discretised by a conforming Galerkin finite-element method. This leads to a linear system of ordinary differential equations (ODEs), see e.g. [10], [20],

$$B\dot{u} + Au = f \ , \quad u(0) = u_0 \ , \quad t > 0 \ , \tag{1.1}$$

with B the symmetric positive definite mass matrix, A the stiffness matrix, and $u(t) = (u_1(t), u_2(t), \dots, u_d(t))^t$ the unknown solution vector.

In [10] we considered solving (1.1) with the continuous-time multigrid waveform relaxation method. This method is based on waveform relaxation, a highly parallel technique for solving very large systems of ODEs [12], [15]. It is accelerated by using multigrid, a very efficient method for solving elliptic partial differential equations, see e.g. [1], [4], [24]. The continuous-time waveform relaxation method differs from standard ODE solvers in that it computes a solution along a continuous time interval. It requires the analytical solution of certain ODEs and the exact continuous representation of certain functions. The method is therefore mainly of theoretical interest. In an actual implementation of the method, the algorithm is replaced by a discrete-time algorithm. That is, functions are represented discretely as vectors defined on successive time levels, and the ODEs are solved by using standard time-stepping techniques.

The resulting *discrete-time multigrid waveform relaxation method* belongs to the class of parabolic multigrid methods. These are multigrid methods for time-dependent problems designed to operate on grids extending in space and time. Other examples of such methods are the time-parallel multigrid method [3], [7] and the space–time multigrid method [8]. These methods are highly efficient on parallel computers, possibly outperforming parallel implementations of standard time-stepping methods by orders of magnitude [9], [23]. Their convergence characteristics as iterative solvers are often similar to the convergence characteristics of multigrid methods for stationary problems, although different parabolic multigrid methods may have very different robustness characteristics. The waveform method, in par-

*Received by the editors May 20, 1994; accepted for publication (in revised form) March 1, 1995.

†Katholieke Universiteit Leuven, Department of Computer Science, Celestijnenlaan 200A, B-3001 Heverlee, Belgium (jan.janssen@cs.kuleuven.ac.be). This text presents research results of the Belgian Incentive Program "Information Technology"—Computer Science of the Future, initiated by the Belgian State–Prime Minister's Office for Science, Technology and Culture. The scientific responsibility is assumed by its authors.

‡California Institute of Technology, Applied Mathematics 217-50, Pasadena, CA 91125 (stefan@ama.caltech.edu). This work was supported in part by the NSF under Cooperative Agreement No. CCR-9120008.

ticular, was shown to be very robust across a wide range of time-discretisation schemes. We refer to [8], [22] for a further discussion.

In this paper, we continue our study of the multigrid waveform relaxation method for systems of the form (1.1). In particular, we analyse the effect of time discretisation when linear multistep formulae are used. The structure of this paper is similar to the structure of [10]. In §2, we analyse the spectral properties of certain operators that arise in the formulation of the waveform relaxation methods. After a brief review of some definitions and properties of linear multistep methods in §3, we investigate the convergence of the discrete-time standard waveform relaxation method (§4) and of its two-grid acceleration (§5), both on finite and infinite time intervals. For systems of the form (1.1) with $B = I$, the discrete-time waveform method and its multigrid variant have been investigated in [14], [16], [17], [21]. Our results are qualitatively very similar, and generalise the ones found in these references. In §6, we perform a model problem analysis for a one-dimensional and two-dimensional model problem. Finally, in §7, extensive numerical results are reported.

2. Spectral properties of a special operator. We will show in §§4 and 5 that the discrete-time waveform relaxation method and its two-grid acceleration can be written as successive approximation schemes of the form

$$u_\tau^{(\nu)} = \mathcal{H}_\tau u_\tau^{(\nu-1)} + \varphi_\tau \,. \tag{2.1}$$

We use subscript τ-notation to denote vectors or sequences, e.g. $u_\tau^{(\nu)} = \{u_i^{(\nu)}\}_{i=0}^{N-1}$, where N is the (possibly infinite) number of components. Each component is a d-vector, and will typically approximate the solution of the system of d differential equations (1.1) at a given time level. Operator $\mathcal{H}_\tau$ is a linear discrete convolution operator with matrix-valued kernel h_τ,

$$(\mathcal{H}_\tau u_\tau)_j = (h_\tau \star u_\tau)_j = \sum_{i=0}^{j} h_{j-i} u_i \,, \quad j = 0, \ldots, N-1 \,.$$

The convergence properties of operator $\mathcal{H}_\tau$ will be analysed in the spaces of $\mathbb{C}^d$-valued p-summable sequences of length N, $l_p(N; \mathbb{C}^d)$, or $l_p(N)$ for short. These are Banach spaces with norms given by

$$||u_\tau||_p = \begin{cases} \sqrt[p]{\sum_{i=0}^{N-1} ||u_i||^p} & 1 \le p < \infty, \\ \sup\limits_{0 \le i < N} \{||u_i||\} & p = \infty, \end{cases} \tag{2.2}$$

with $||\cdot||$ any usual $\mathbb{C}^d$ vector norm. Recall that the iterative scheme (2.1) is convergent if and only if the spectral radius of $\mathcal{H}_\tau$, denoted by $\rho(\mathcal{H}_\tau)$, is smaller than one. The spectral radius is defined as the largest value ρ for which $|\lambda| > \rho$ implies that $\lambda - \mathcal{H}_\tau$ has a bounded inverse. When N is finite, it equals the magnitude of the largest eigenvalue of $\mathcal{H}_\tau$.

2.1. Spectral radius on finite time intervals.

LEMMA 2.1. *Consider $\mathcal{H}_\tau$ as an operator in $l_p(N)$, with $1 \le p \le \infty$ and N finite. Then, $\mathcal{H}_\tau$ is a bounded operator and*

$$\rho(\mathcal{H}_\tau) = \rho(h_0) = \rho(\mathbf{H}_\tau(\infty)) \,, \tag{2.3}$$

with $\mathbf{H}_\tau(z) = \sum_{i=0}^{N-1} h_i z^{-i}$ the discrete Laplace transform of h_τ.

Proof. Since $\mathcal{H}_\tau$ is a linear operator in a finite-dimensional space, boundedness of $\mathcal{H}_\tau$ follows. The operation $\mathcal{H}_\tau u_\tau$ can be represented in standard linear algebra notation as a matrix-vector product,

$$\begin{bmatrix} h_0 & & & & & \\ h_1 & h_0 & & & & \\ h_2 & h_1 & h_0 & & & \\ \cdot & \cdot & \cdot & \cdot & & \\ \cdot & \cdot & \cdot & \cdot & \cdot & \\ h_{N-1} & \cdot & \cdot & h_2 & h_1 & h_0 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ \cdot \\ \cdot \\ \cdot \\ u_{N-1} \end{bmatrix} . \tag{2.4}$$

The spectral radius of operator $\mathcal{H}_\tau$ equals the spectral radius of the $N \times N$-block lower triangular Toeplitz matrix in (2.4). By consequence, $\rho(\mathcal{H}_\tau) = \rho(h_0)$. The second equality follows immediately. □

2.2. Spectral radius on infinite time intervals.

LEMMA 2.2. *Suppose $h_\tau \in l_1(\infty)$, and consider $\mathcal{H}_\tau$ as an operator in $l_p(\infty)$, with $1 \le p \le \infty$. Then, $\mathcal{H}_\tau$ is bounded and*

$$\rho(\mathcal{H}_\tau) = \max_{|z|\ge 1} \rho(\mathbf{H}_\tau(z)) \tag{2.5}$$

$$= \max_{|z|=1} \rho(\mathbf{H}_\tau(z)) , \tag{2.6}$$

with $\mathbf{H}_\tau(z) = \sum_{i=0}^{\infty} h_i z^{-i}$ the discrete Laplace transform of h_τ.

The outline of our proof is very similar to the one given in [16, Thm. 3.1]. Yet, here, it is phrased in terms of general convolution operators. A similar line of arguments is implied in the proof of [14, Prop. 9]. The proof is based on the discrete version of the Paley–Wiener theorem [13]. This theorem states that the solution of a discrete Volterra convolution equation $x_\tau + h_\tau \star x_\tau = f_\tau$, with $f_\tau \in l_p(\infty)$ and $h_\tau \in l_1(\infty)$ is bounded in $l_p(\infty)$ if and only if $\det(I + \mathbf{H}_\tau(z)) \neq 0$ for $|z| \ge 1$, with $\mathbf{H}_\tau(z)$ the discrete Laplace transform of h_τ.

Proof. The boundedness of $\mathcal{H}_\tau$ follows from the fact that $l_1 \star l_p \subset l_p$. Indeed, applying Young's inequality for discrete convolution products [6, p. 198] yields

$$||\mathcal{H}_\tau u_\tau||_p \le ||h_\tau||_1 \, ||u_\tau||_p .$$

By definition, the spectral radius of $\mathcal{H}_\tau$ is the smallest value of ρ for which $|\lambda| > \rho$ implies that $\lambda - \mathcal{H}_\tau$ has a bounded inverse in $l_p(\infty)$. Consider

$$\lambda u_\tau - \mathcal{H}_\tau u_\tau = \lambda u_\tau - h_\tau \star u_\tau = f_\tau ,$$

with $f_\tau \in l_p(\infty)$. Suppose $\lambda \neq 0$, then this can be rewritten as a convolution equation

$$u_\tau - \frac{1}{\lambda} h_\tau \star u_\tau = \frac{1}{\lambda} f_\tau .$$

By the Paley–Wiener theorem, it follows that u_τ is bounded if and only if

$$\det\left(I - \frac{1}{\lambda}\mathbf{H}_\tau(z)\right) \neq 0 \quad \text{for} \quad |z| \ge 1 ,$$

or, equivalently,

$$\rho(\mathcal{H}_\tau) = \sup_{|z|\ge 1} \rho(\mathbf{H}_\tau(z)) .$$

Note that $\mathbf{H}_\tau(z)$ is analytic for $|z| > 1$, including $z = \infty$, and, since $h_\tau \in l_1(\infty)$, it is continuous for $|z| \geq 1$. Also, the spectral radius satisfies the maximum principle. Hence, we obtain (2.5) and (2.6). □

Remark 2.1. In the case of $d = 1$, this lemma corresponds to a well-known spectral property of semi-infinite Toeplitz operators [18, Thm. 2.1].

In $l_2(\infty)$, an analogous result holds for the norm.

LEMMA 2.3. *Suppose $h_\tau \in l_1(\infty)$, and consider $\mathcal{H}_\tau$ as an operator in $l_2(\infty)$. Denote by $||\cdot||_2$ the l_2-norm (2.2) with $||\cdot||$ the standard Euclidean vector norm. Then,*

$$||\mathcal{H}_\tau||_2 = \max_{|z|\geq 1} ||\mathbf{H}_\tau(z)|| \tag{2.7}$$

$$= \max_{|z|=1} ||\mathbf{H}_\tau(z)||\,, \tag{2.8}$$

with $\mathbf{H}_\tau(z)$ the discrete Laplace transform of h_τ.

Proof. The proof is based on Parseval's relation for vector-valued l_2-sequences,

$$||u_\tau||_2 := ||\{u_i\}_{i=0}^{\infty}||_2 = \left\| \sum_{i=0}^{\infty} u_i z^{-i} \right\|_{H_2} ,$$

where $||\cdot||_{H_2}$ is the norm in the Hardy–Lebesgue space of square integrable functions analytic outside the unit disk,

$$||f(z)||_{H_2} = \sup_{r>1} \left(\frac{1}{2\pi} \int_0^{2\pi} ||f(re^{i\theta})||^2 d\theta \right)^{1/2} .$$

The Parseval relation for the scalar case can be found, e.g., in [25, p. 41]. By definition of operator norm and by Parseval's relation, we have

$$||\mathcal{H}_\tau||_2 = \sup \frac{||\mathcal{H}_\tau u_\tau||_2}{||u_\tau||_2} = \sup \frac{||\mathbf{H}_\tau(z)\tilde{u}_\tau(z)||_{H_2}}{||\tilde{u}_\tau(z)||_{H_2}} ,$$

with $\tilde{u}_\tau(z)$ the discrete Laplace transform of u_τ. $||\mathcal{H}_\tau||_2$ can be seen to be equal to $\sup_{|z|\geq 1} ||\mathbf{H}_\tau(z)||$. (For the technical details of this last step, we refer to the proof of a very similar theorem, [2, Thm. 2.2], which deals with operator norms of Fourier multipliers.) Consideration of the analyticity and continuity of $\mathbf{H}_\tau(z)$ leads to (2.7) and (2.8). □

Remark 2.2. From (2.3) and (2.5), it follows that the spectral radius of $\mathcal{H}_\tau$ on finite time intervals is smaller than the spectral radius of $\mathcal{H}_\tau$ on infinite time intervals.

3. Some linear multistep formulae.

3. Some linear multistep formulae. For the reader's convenience, we recall the general linear multistep formula for calculating the solution to the ODE $\dot{y} = f(t, y)$ with $y(0) = y_0$, see e.g. [11, p. 11],

$$\frac{1}{\tau} \sum_{j=0}^{k} \alpha_j y_{n+j} = \sum_{j=0}^{k} \beta_j f_{n+j} \,. \tag{3.1}$$

In this formula, α_j and β_j are real constants, and τ denotes a constant step size. We shall assume that k starting values $y_0, y_1, \ldots, y_{k-1}$ are given.

DEFINITION 3.1. *The characteristic polynomials of the linear multistep method are given by*

$$a(z) = \sum_{j=0}^{k} \alpha_j z^j \quad \textit{and} \quad b(z) = \sum_{j=0}^{k} \beta_j z^j \,.$$

Throughout this paper we adhere to some common assumptions. The linear multistep method is irreducible: $a(z)$ and $b(z)$ have no common roots; the linear multistep method is consistent: $a(1) = 0$ and $a'(1) = b(1)$; the linear multistep method is zero-stable: all roots of $a(z)$ are inside the closed unit disk and every root with modulus one is simple. For future reference, we also define the stability region of a linear multistep method, and the related notion of $A(\alpha)$-stability, see e.g. [5], [11].

DEFINITION 3.2. *The stability region S consists of those $\mu \in \bar{\mathbb{C}}$ for which the polynomial $a(z) - \mu b(z)$ (around $\mu = \infty$: $\mu^{-1}a(z) - b(z)$) satisfies the root condition: all roots satisfy $|z_j| \leq 1$ and those of modulus one are simple.*

DEFINITION 3.3. *A multistep method is called*

(i) *$A(\alpha)$-stable, $0 < \alpha < \frac{\pi}{2}$, if $S \supset \Sigma_\alpha = \{z : |\mathrm{Arg}(-z)| < \alpha,\ z \neq 0\}$*

(ii) *A-stable if S contains the left-half complex plane.*

4. The waveform relaxation method. The continuous-time waveform relaxation method for solving initial value problem (1.1) is defined by the splittings $B = M_B - N_B$, $A = M_A - N_A$, and the iteration scheme

$$M_B \dot{u}^{(\nu)} + M_A u^{(\nu)} = N_B \dot{u}^{(\nu-1)} + N_A u^{(\nu-1)} + f\ , \tag{4.1}$$

with $u^{(\nu)}(0) = u_0$. We assume the splitting is such that M_B is invertible. This iterative scheme can be written in explicit form as $u^{(\nu)} = \mathcal{K}u^{(\nu-1)} + \varphi$. The convergence properties of iteration operator $\mathcal{K}$, the *continuous-time waveform relaxation operator*, have been studied in [10]. They are expressed in terms of the matrix

$$\mathbf{K}(z) = (zM_B + M_A)^{-1}(zN_B + N_A)\ . \tag{4.2}$$

It was shown on finite and infinite time intervals, respectively, and with $i = \sqrt{-1}$, that

$$\rho(\mathcal{K}) = \rho(\mathbf{K}(\infty)) \quad \text{and} \quad \rho(\mathcal{K}) = \sup_{\mathrm{Re}(z)\geq 0} \rho(\mathbf{K}(z)) = \sup_{\xi\in\mathbb{R}} \rho(\mathbf{K}(i\xi))\ . \tag{4.3}$$

4.1. The discrete-time waveform relaxation operator. Application of linear multistep formula (3.1) to the continuous-time iteration scheme (4.1) leads to

$$\begin{aligned} &\frac{1}{\tau}\sum_{j=0}^{k} \alpha_j M_B u_{n+j}^{(\nu)} + \sum_{j=0}^{k} \beta_j M_A u_{n+j}^{(\nu)} \\ &\quad = \frac{1}{\tau}\sum_{j=0}^{k} \alpha_j N_B u_{n+j}^{(\nu-1)} + \sum_{j=0}^{k} \beta_j N_A u_{n+j}^{(\nu-1)} + \sum_{j=0}^{k} \beta_j f_{n+j}\ , \quad n \geq 0\ . \end{aligned} \tag{4.4}$$

We do not iterate on the k starting values, i.e., $u_j^{(\nu)} = u_j^{(\nu-1)} = u_j$, for $j < k$. In the remainder of the text we shall concentrate on the use of implicit methods, i.e., $\beta_k \neq 0$. Equation (4.4) can then be solved uniquely for every n if and only if the following condition is satisfied:

$$\frac{\alpha_k}{\beta_k} \notin \sigma\left(-\tau M_B^{-1} M_A\right)\ , \tag{4.5}$$

where $\sigma(\cdot)$ denotes the spectrum. Further on we shall refer to this condition as the discrete solvability condition.

Iteration (4.4) can be rewritten as $u_\tau^{(\nu)} = \mathcal{K}_\tau u_\tau^{(\nu-1)} + \varphi_\tau$. Because we do not iterate on the starting values, we use a slightly different subscript τ-notation here than the one in (2.1), that is,

$$u_\tau = \{u_{k+i}\}_{i=0}^{N-1}\ . \tag{4.6}$$

(Alternatively, we could have used negative indices to denote the time levels associated with the k starting values, as is done in [13], [14]. This, however, would require some shifting in the indices of formulae (3.1) and (4.4).) The precise expression for φ_τ can be calculated following the lines of [17, p. 536–537]. It depends on the values of $f_n, n \geq 0$ and on the starting values u_n, $n < k$. In order to determine the nature of $\mathcal{K}_\tau$, the *discrete-time waveform relaxation operator*, we rewrite (4.4) using $e_n^{(\nu)} = u_n^{(\nu)} - u_n$. Here, u_n is the exact solution of ODE (1.1) when discretised using the linear multistep method. This gives

$$\frac{1}{\tau}\sum_{j=0}^{k}\alpha_j M_B e_{n+j}^{(\nu)} + \sum_{j=0}^{k}\beta_j M_A e_{n+j}^{(\nu)} = \frac{1}{\tau}\sum_{j=0}^{k}\alpha_j N_B e_{n+j}^{(\nu-1)} + \sum_{j=0}^{k}\beta_j N_A e_{n+j}^{(\nu-1)}\ , \quad n \geq 0\,.$$

With $C_j = \frac{1}{\tau}\alpha_j M_B + \beta_j M_A$, and $D_j = \frac{1}{\tau}\alpha_j N_B + \beta_j N_A$, this becomes

$$\sum_{j=0}^{k} C_j e_{n+j}^{(\nu)} = \sum_{j=0}^{k} D_j e_{n+j}^{(\nu-1)}\ , \quad n \geq 0\,. \tag{4.7}$$

Note that $e_j^{(\nu)} = e_j^{(\nu-1)} = 0$, $j < k$. When we combine the first N equations, i.e., the equations for the unknowns on time steps $k, \ldots, N+k-1$, and after introducing vector $E^{(\nu)} = [e_k^{(\nu)} e_{k+1}^{(\nu)} \ldots e_{N+k-1}^{(\nu)}]^t$, we get

$$E^{(\nu)} = C^{-1} D\, E^{(\nu-1)}\,. \tag{4.8}$$

Matrices C and D are $N \times N$-block lower triangular matrices with $k+1$ constant diagonals. The blocks on the jth diagonal are given respectively by C_{k-j} and D_{k-j}. It follows immediately that matrix $C^{-1}D$ is a $N \times N$-block lower triangular Toeplitz matrix. Hence, $\mathcal{K}_\tau$ is a discrete linear convolution operator on the l_p-space of vectors or sequences of length N. The jth component of the matrix-valued discrete convolution kernel k_τ equals the (constant) submatrix on the jth lower block diagonal of $C^{-1}D$.

In the theory we shall need the discrete Laplace transform of the convolution kernel. It can be found by discrete Laplace-transforming equation (4.7). If $\tilde{e}_\tau^{(\nu)}(z)$ denotes the transform of $e_\tau^{(\nu)}$, we obtain

$$\tilde{e}_\tau^{(\nu)}(z) = \mathbf{K}_\tau(z)\tilde{e}_\tau^{(\nu-1)}(z)\,,$$

with the discrete-time waveform relaxation matrix given by

$$\mathbf{K}_\tau(z) = (a(z)M_B + \tau b(z)M_A)^{-1}(a(z)N_B + \tau b(z)N_A)\,. \tag{4.9}$$

By comparison to (4.2) the following relation results:

$$\mathbf{K}_\tau(z) = \mathbf{K}\left(\frac{1}{\tau}\frac{a}{b}(z)\right)\,. \tag{4.10}$$

Note that (4.10) still holds when $\frac{a}{b}(z)$ is set to ∞ in the case of $b(z) = 0$. (In this case $a(z) \neq 0$, since the characteristic polynomials have no common roots.)

4.2. Convergence analysis.

4.2.1. Convergence on finite time intervals.

THEOREM 4.1. *Assume that condition* (4.5) *is satisfied, and consider $\mathcal{K}_\tau$ as an operator in $l_p(N)$, with $1 \leq p \leq \infty$ and N finite. Then, $\mathcal{K}_\tau$ is bounded and*

$$\rho(\mathcal{K}_\tau) = \rho\left(\mathbf{K}\left(\frac{1}{\tau}\frac{\alpha_k}{\beta_k}\right)\right)\,. \tag{4.11}$$

Proof. The theorem follows from Lemma 2.1 and the observation that

$$\lim_{z\to\infty} \mathbf{K}_\tau(z) = \lim_{z\to\infty} \mathbf{K}\left(\frac{1}{\tau}\frac{a}{b}(z)\right) = \mathbf{K}\left(\frac{1}{\tau}\frac{\alpha_k}{\beta_k}\right). \qquad \square$$

4.2.2. Convergence on infinite time intervals. The following lemma deals with the boundedness of the discrete-time waveform relaxation operator $\mathcal{K}_\tau$. It is proved using a matrix-valued version of Wiener's inversion theorem [13, p. 446] and [16, p. 577], which is stated here for the reader's convenience.

THEOREM 4.2 (WIENER'S INVERSION THEOREM). *Given a matrix-valued sequence A_τ such that $A_\tau \in l_1(\infty)$, assume that*

$$\det \sum_{i=0}^{\infty} A_i z^{-i} \neq 0$$

for $|z| \geq 1$. Setting $\sum_{i=0}^{\infty} B_i z^{-i} = \left(\sum_{i=0}^{\infty} A_i z^{-i}\right)^{-1}$, we have $B_\tau \in l_1(\infty)$.

LEMMA 4.3. *If $\sigma(-\tau M_B^{-1} M_A) \subset \operatorname{int} S$, then $\mathcal{K}_\tau$ is bounded in $l_p(\infty)$.*

Proof. It is sufficient to prove that the kernel k_τ of the discrete convolution operator $\mathcal{K}_\tau$ is an l_1-sequence. To this end, consider first the l_1-sequence

$$\alpha_k M_B + \tau\beta_k M_A,\ \alpha_{k-1} M_B + \tau\beta_{k-1} M_A, \ldots, \alpha_0 M_B + \tau\beta_0 M_A,\ 0,\ 0, \ldots.$$

Its discrete Laplace transform equals the matrix function $z^{-k}(a(z)M_B + \tau b(z)M_A)$. By Wiener's theorem, we have that the inverse, $(a(z)M_B + \tau b(z)M_A)^{-1} z^k$, is the transform of another l_1-sequence, say r_τ, if

$$\det(a(z)M_B + \tau b(z)M_A) \neq 0 \text{ for } |z| \geq 1. \tag{4.12}$$

Next, consider the l_1-sequence

$$s_\tau = \alpha_k N_B + \tau\beta_k N_A,\ \alpha_{k-1} N_B + \tau\beta_{k-1} N_A, \ldots, \alpha_0 N_B + \tau\beta_0 N_A,\ 0,\ 0, \ldots,$$

the discrete Laplace transform of which is given by $z^{-k}(a(z)N_B + \tau b(z)N_A)$. The convolution of r_τ and s_τ is another l_1-sequence, which can be seen to be equal to the kernel k_τ. Indeed, the discrete Laplace transform of $r_\tau \star s_\tau$ is identical to $\mathbf{K}_\tau(z)$. As a result, it follows that $\mathcal{K}_\tau$ is bounded if (4.12) is satisfied.

Suppose there is a z with $|z| \geq 1$ such that

$$\det(a(z)M_B + \tau b(z)M_A) = 0. \tag{4.13}$$

Then necessarily $b(z) \neq 0$. (If $b(z) = 0$ then $a(z) \neq 0$, because $a(z)$ and $b(z)$ have no common roots. Since M_B is assumed to be invertible, equality (4.13) cannot hold.) Hence, we obtain

$$\det\left(\frac{a}{b}(z)M_B + \tau M_A\right) = 0,$$

and therefore $\frac{a}{b}(z) \in \sigma(-\tau M_B^{-1} M_A)$. Since $|z| \geq 1$, it follows that $-\tau M_B^{-1} M_A$ has an eigenvalue which is not an interior point of S. This contradicts the assumption of the lemma. Hence, (4.12) is satisfied. $\square$

Remark 4.1. Condition $\sigma(-\tau M_B^{-1} M_A) \subset \operatorname{int} S$ implies the discrete solvability condition (4.5). Indeed, since $\frac{\alpha_k}{\beta_k} = \frac{a}{b}(\infty)$, it follows that $\frac{\alpha_k}{\beta_k} \notin \operatorname{int} S$, and, therefore, $\frac{\alpha_k}{\beta_k} \notin \sigma(-\tau M_B^{-1} M_A)$.

Remark 4.2. Condition $\sigma(-\tau M_B^{-1} M_A) \subset \operatorname{int} S$ implies that all poles of $\mathbf{K}(z)$ are in the interior of the scaled stability region $\frac{1}{\tau} S$.

THEOREM 4.4. *Assume* $\sigma(-\tau M_B^{-1} M_A) \subset \operatorname{int} S$, *and consider* $\mathcal{K}_\tau$ *as an operator in* $l_p(\infty)$, *with* $1 \leq p \leq \infty$. *Then,*

$$\rho(\mathcal{K}_\tau) = \sup\{\rho(\mathbf{K}(z)) | \tau z \in \mathbb{C} \setminus \operatorname{int} S\} \tag{4.14}$$

$$= \sup_{\tau z \in \partial S} \rho(\mathbf{K}(z)) \,. \tag{4.15}$$

Proof. As $\sigma(-\tau M_B^{-1} M_A) \subset \operatorname{int} S$, it follows that $k_\tau \in l_1(\infty)$. Lemma 2.2 yields

$$\rho(\mathcal{K}_\tau) = \max_{|z|\geq 1} \rho(\mathbf{K}_\tau(z)) = \max_{|z|\geq 1} \rho\left(\mathbf{K}\left(\frac{1}{\tau}\frac{a}{b}(z)\right)\right) .$$

By definition of the stability region,

$$\bar{\mathbb{C}} \setminus \operatorname{int} S = \left\{\frac{a}{b}(z) : |z| \geq 1\right\} ,$$

and thereby (4.14) follows. Equality (4.15) is obtained by the maximum principle. Note that we write "sup" instead of "max," since the maximum may be approached at infinity. □

In $l_2(\infty)$, a similar result holds for the norm by application of Lemma 2.3.

THEOREM 4.5. *Assume* $\sigma(-\tau M_B^{-1} M_A) \subset \operatorname{int} S$, *and consider* $\mathcal{K}_\tau$ *as an operator in* $l_2(\infty)$. *Denote by* $||\cdot||_2$ *the* l_2*-norm* (2.2) *with* $||\cdot||$ *the standard Euclidean vector norm. Then,*

$$||\mathcal{K}_\tau||_2 = \sup\{||\mathbf{K}(z)|| : \tau z \in \mathbb{C} \setminus \operatorname{int} S\} \tag{4.16}$$

$$= \sup_{\tau z \in \partial S} ||\mathbf{K}(z)|| \,. \tag{4.17}$$

Analogous to the discussion in [17, Thm. 4.2], we can make the following note.

Remark 4.3. When the assumption in the above theorems is violated, a weaker condition may be satisfied: $\sigma(-\tau M_B^{-1} M_A) \subset \operatorname{int} S_{\gamma\tau}$, where $S_{\gamma\tau}$ consists of all μ for which $a(e^{-\gamma\tau}z) - \mu b(e^{-\gamma\tau}z)$ (around $\mu = \infty$: $\mu^{-1}a(e^{-\gamma\tau}z) - b(e^{-\gamma\tau}z)$) satisfies the root condition. The analysis then can be redone using an exponentially scaled norm,

$$||u_\tau||_\gamma = ||\{u_i\}||_\gamma = ||\{e^{-\gamma\tau i} u_i\}|| \,. \tag{4.18}$$

The norm in the right-hand side is a standard p-norm (2.2). With this change of norm, the suprema in Theorems 4.4 and 4.5 have to be taken over all τz in $\mathbb{C} \setminus \operatorname{int} S_{\gamma\tau}$, or, after application of the maximum principle, over $\partial S_{\gamma\tau}$.

4.3. Discrete-time versus continuous-time results. The continuous-time results (4.3) are regained when we let $\tau \to 0$ in the convergence formulae for operator $\mathcal{K}_\tau$. For finite time intervals, we have

$$\lim_{\tau\to 0} \rho(\mathcal{K}_\tau) = \lim_{\tau\to 0} \rho\left(\mathbf{K}\left(\frac{1}{\tau}\frac{\alpha_k}{\beta_k}\right)\right) = \rho(\mathbf{K}(\infty)) = \rho(\mathcal{K}) .$$

A similar result is found for infinite time intervals. Note that the tangent to ∂S in the origin of the complex plane is the imaginary axis, for any consistent linear multistep method. As such, the boundary of the scaled stability region $\partial(\frac{1}{\tau} S)$ tends to the imaginary axis when $\tau \to 0$. Consequently,

$$\lim_{\tau\to 0} \rho(\mathcal{K}_\tau) = \lim_{\tau\to 0} \sup_{\tau z \in \partial S} \rho(\mathbf{K}(z)) = \sup_{\xi \in \mathbb{R}} \rho(\mathbf{K}(i\xi)) = \rho(\mathcal{K}) .$$

Furthermore, for a fixed time step τ, we can prove the following theorem for $A(\alpha)$-stable linear multistep methods (see Definition 3.3). The theorem is closely related to [14, Prop. 9], where multigrid waveform relaxation on finite-difference grids is analysed. We reformulate the proof, using our notations, for completeness.

THEOREM 4.6. *Assume $\sigma(-\tau M_B^{-1}M_A) \subset \Sigma_\alpha$. Consider $\mathcal{K}_\tau$ as an operator in $l_p(\infty)$ and $\mathcal{K}$ as an operator in $L_p(0,\infty)$, with $1 \leq p \leq \infty$. Then,*

(i) *if the linear multistep method is A-stable, then $\rho(\mathcal{K}_\tau) \leq \rho(\mathcal{K})$;*

(ii) *if the linear multistep method is $A(\alpha)$-stable, then*

$$\rho(\mathcal{K}_\tau) \leq \sup_{z\in\Sigma_\alpha^c} \rho(\mathbf{K}(z)) = \sup_{z\in\partial\Sigma_\alpha^c} \rho(\mathbf{K}(z)) \,, \tag{4.19}$$

with $\Sigma_\alpha^c = \mathbb{C} \setminus \Sigma_\alpha = \{z : |\text{Arg}(z)| \leq \pi - \alpha\}$.

Proof. Part (i) is a special case of (ii) with $\alpha = \pi/2$, combined with the second equality of (4.3). For part (ii), we notice that we may apply Theorem 4.4 since $\sigma(-\tau M_B^{-1}M_A) \subset \Sigma_\alpha \subset$ int S. Therefore,

$$\rho(\mathcal{K}_\tau) = \max_{|z|\geq 1} \rho\left(\mathbf{K}\left(\frac{1}{\tau}\frac{a}{b}(z)\right)\right) . \tag{4.20}$$

If the multistep method is $A(\alpha)$-stable, then $\frac{a}{b}(z) \in \Sigma_\alpha^c$ for $|z| \geq 1$. Combining the latter with (4.20) yields the inequality of (4.19). The equality is obtained by the maximum principle. □

5. The multigrid waveform relaxation method. The splittings of matrices B and A used in actual computations typically correspond to Gauss–Seidel or weighted Jacobi splittings. Each iteration defined by (4.1) can then be computed as the solution of d ordinary differential equations, each in a single unknown. The resulting iteration can be accelerated by using the multigrid principle, in a very similar way as the standard pointwise relaxation methods are accelerated when solving elliptic partial differential equations.

The continuous-time two-grid waveform relaxation scheme is sketched below. We refer to [10] for a more elaborate description. The algorithm uses two nested grids, a coarse grid Ω_H and a fine grid Ω_h. Grid functions are mapped from the one grid to the other by a prolongation (or interpolation) operator ($p : \Omega_H \to \Omega_h$) and a restriction operator ($r : \Omega_h \to \Omega_H$). The discretisation on the fine grid is defined by the matrices B_h and A_h, the discretisation on the coarse grid by B_H and A_H. One iteration transforms iterate $u^{(\nu-1)}$ into $u^{(\nu)}$ in three steps.

(i) Presmoothing. Set $x_h^{(0)} = u^{(\nu-1)}$ and perform ν_1 fine-grid waveform relaxation steps: for $\nu = 1, 2, \ldots, \nu_1$, solve

$$M_{B_h}\dot{x}_h^{(\nu)} + M_{A_h}x_h^{(\nu)} = N_{B_h}\dot{x}_h^{(\nu-1)} + N_{A_h}x_h^{(\nu-1)} + f_h \ , \ \text{with} \ x_h^{(\nu)}(0) = u_0 \, . \tag{5.1}$$

(ii) Coarse-grid correction. Calculate the defect

$$d_h = B\dot{x}_h^{(\nu_1)} + A_h x_h^{(\nu_1)} - f_h \, .$$

Solve the coarse-grid defect equation

$$B_H\dot{v}_H + A_H v_H = rd_h, \ \text{with} \ v_H(0) = 0 \, ,$$

and correct,

$$\bar{x}_h = x_h^{(\nu_1)} - pv_H \, .$$

(iii) Postsmoothing. Set $x_h^{(0)} = \bar{x}_h$ and perform ν_2 fine-grid waveform relaxation steps (5.1). Set $u^{(\nu)} = x_h^{(\nu_2)}$.

This two-grid cycle can be written as $u^{(\nu)} = \mathcal{M}u^{(\nu-1)} + \varphi$, where $\mathcal{M}$ is called the *continuous-time two-grid waveform relaxation operator*. The convergence formulae of $\mathcal{M}$ as an iteration operator resemble those of the standard waveform relaxation method. More precisely, in [10] we find for the finite and for the infinite time-interval case, respectively,

$$(5.2) \qquad \rho(\mathcal{M}) = \rho(\mathbf{M}(\infty)) \quad \text{and} \quad \rho(\mathcal{M}) = \sup_{\mathrm{Re}(z)\geq 0} \rho(\mathbf{M}(z)) = \sup_{\xi\in\mathbb{R}} \rho(\mathbf{M}(i\xi)) \,.$$

$\mathbf{M}(z)$, the continuous-time two-grid waveform relaxation matrix, is given by

$$\mathbf{M}(z) = \mathbf{K}^{\nu_2}(z)(I - p(zB_H + A_H)^{-1}r(zB_h + A_h))\mathbf{K}^{\nu_1}(z) \,,$$

with $\mathbf{K}(z)$ the fine-grid matrix given in (4.2). We recall from [10, Rem. 4.1] the following important remark.

Remark 5.1. In the case of a Gauss–Seidel (or weighted Jacobi) splitting of A_h and B_h, $\mathbf{K}(z)$ and $\mathbf{M}(z)$ are respectively the Gauss–Seidel (or weighted Jacobi) iteration matrix and the two-grid iteration matrix for the system $(zB_h + A_h)u_h = f_h$.

In the following, the discrete-time variant of this *two-grid* waveform relaxation method is theoretically investigated. We refer to the Appendix for a similar convergence study of the *multigrid* waveform relaxation method. The latter is defined by solving the coarse-grid defect equation using one or more similar two-grid waveform relaxation cycles, and applying this idea recursively.

5.1. The discrete-time two-grid waveform relaxation operator. We discretise the equations of the continuous-time two-grid cycle using a linear multistep method with a fixed time step τ. As before, we assume that we do not iterate on the k given starting values. The discrete-time two-grid cycle defines a linear operator $\mathcal{M}_\tau$, which satisfies

$$(5.3) \qquad u_\tau^{(\nu)} = \mathcal{M}_\tau u_\tau^{(\nu-1)} + \varphi_\tau \quad \text{and} \quad e_\tau^{(\nu)} = \mathcal{M}_\tau e_\tau^{(\nu-1)} \,,$$

where $e_\tau^{(\nu)}$ is the error of the νth iterate. Our notation is again similar to (4.6). $\mathcal{M}_\tau$ is called the *discrete-time two-grid waveform relaxation operator*.

The second equation of (5.3) can be reformulated similar to how we arrived at (4.8),

$$(5.4) \qquad E^{(\nu)} = (C_h^{-1}D_h)^{\nu_2}(I - PF_H^{-1}RF_h)(C_h^{-1}D_h)^{\nu_1}E^{(\nu-1)} \,.$$

Here, $E^{(\nu)} = [e_k^{(\nu)}e_{k+1}^{(\nu)} \dots e_{N+k-1}^{(\nu)}]^t$. Matrices C_h, D_h, F_H and F_h are $N \times N$-block lower triangular matrices with $k+1$ constant diagonals. The blocks of the jth diagonal equal $(C_h)_{k-j}$, $(D_h)_{k-j}$, $(F_H)_{k-j}$, and $(F_h)_{k-j}$, respectively, with

$$(C_h)_j = \frac{1}{\tau}\alpha_j M_{B_h} + \beta_j M_{A_h} \,, \quad (D_h)_j = \frac{1}{\tau}\alpha_j N_{B_h} + \beta_j N_{A_h} \,,$$

and

$$(F_H)_j = \frac{1}{\tau}\alpha_j B_H + \beta_j A_H \,, \quad (F_h)_j = \frac{1}{\tau}\alpha_j B_h + \beta_j A_h \,.$$

Matrices P and R are block diagonal with constant diagonal blocks equal to matrices p and r, respectively. I is the identity matrix of dimension $d \times N$. The resulting discrete-time two-grid cycle is well defined, if and only if the following conditions hold:

$$(5.5) \qquad \frac{\alpha_k}{\beta_k} \notin \sigma(-\tau M_{B_h}^{-1}M_{A_h}) \quad \text{and} \quad \frac{\alpha_k}{\beta_k} \notin \sigma(-\tau B_H^{-1}A_H) \,.$$

We shall refer to (5.5) as the discrete solvability conditions for the two-grid algorithm.

It can be seen that the matrix premultiplying $E^{(\nu-1)}$ in (5.4) is a lower triangular block Toeplitz matrix. This implies that $\mathcal{M}_\tau$ is a discrete linear convolution operator. The discrete Laplace transform of its matrix-valued kernel can be found by transforming the equations of the discrete-time two-grid cycle. It is denoted by $\mathbf{M}_\tau(z)$, the discrete-time two-grid waveform relaxation matrix, and equals

$$\mathbf{M}_\tau(z) = \mathbf{K}_\tau{}^{\nu_2}(z)\,\mathbf{C}_\tau(z)\,\mathbf{K}_\tau{}^{\nu_1}(z)\,,$$

with $\mathbf{K}_\tau(z)$ given by (4.9) and $\mathbf{C}_\tau(z)$ given by

$$I - p\,(a(z)B_H + \tau b(z)A_H)^{-1}\,r\,(a(z)B_h + \tau b(z)A_h)\,.$$

Matrix $\mathbf{M}_\tau(z)$ satisfies a similar relation as $\mathbf{K}_\tau(z)$ does in (4.10):

$$\mathbf{M}_\tau(z) = \mathbf{M}\left(\frac{1}{\tau}\frac{a}{b}(z)\right)\,. \tag{5.6}$$

5.2. Convergence analysis. The convergence analysis of operator $\mathcal{M}_\tau$ is very similar to the convergence analysis of the standard waveform relaxation operator $\mathcal{K}_\tau$.

5.2.1. Convergence on finite time intervals.

THEOREM 5.1. *Assume that conditions* (5.5) *are satisfied, and consider $\mathcal{M}_\tau$ as an operator in $l_p(N)$, with $1 \le p \le \infty$ and N finite. Then, $\mathcal{M}_\tau$ is bounded and*

$$\rho(\mathcal{M}_\tau) = \rho\left(\mathbf{M}\left(\frac{1}{\tau}\frac{\alpha_k}{\beta_k}\right)\right)\,. \tag{5.7}$$

Proof. The theorem follows from Lemma 2.1 and (5.6):

$$\rho(\mathcal{M}_\tau) = \rho(\mathbf{M}_\tau(\infty)) = \rho\left(\mathbf{M}\left(\frac{1}{\tau}\frac{a}{b}(\infty)\right)\right) = \rho\left(\mathbf{M}\left(\frac{1}{\tau}\frac{\alpha_k}{\beta_k}\right)\right)\,. \qquad \square$$

5.2.2. Convergence on infinite time intervals. We first prove the boundedness of $\mathcal{M}_\tau$, i.e., we prove the two-grid equivalent of Lemma 4.3.

LEMMA 5.2. *Assume $\sigma(-\tau M_{B_h}^{-1}M_{A_h}) \cup \sigma(-\tau B_H^{-1}A_H) \subset \operatorname{int} S$. Then, $\mathcal{M}_\tau$ is bounded in $l_p(\infty)$.*

Proof. It is sufficient to prove that the kernel of $\mathcal{M}_\tau$ belongs to $l_1(\infty)$. We shall analyse each of the factors in the formula for $\mathbf{M}_\tau(z)$ separately.

We have, from the proof of Lemma 4.3, that $\mathbf{K}_\tau(z)$ is the discrete Laplace transform of an l_1-sequence, say q_τ, if

$$\det\left(a(z)M_{B_h} + \tau b(z)M_{A_h}\right) \neq 0\,,\quad |z| \ge 1\,. \tag{5.8}$$

Consider the l_1-sequence

$$\alpha_k B_H + \tau\beta_k A_H,\ \alpha_{k-1}B_H + \tau\beta_{k-1}A_H, \ldots, \alpha_0 B_H + \tau\beta_0 A_H,\ 0,\ 0, \ldots.$$

Its transform is given by $z^{-k}(a(z)B_H + \tau b(z)A_H)$. By Wiener's inversion theorem, $(a(z)B_H + \tau b(z)A_H)^{-1}z^k$ is the transform of an l_1-sequence, say w_τ, if

$$\det(a(z)B_H + \tau b(z)A_H) \neq 0\,,\quad |z| \ge 1\,. \tag{5.9}$$

Next, consider the l_1-sequences

$$\begin{aligned} i_\tau &= I,\ 0, \ldots, 0,\ 0,\ 0, \ldots,\\ v_\tau &= \alpha_k B_h + \tau\beta_k A_h,\ \alpha_{k-1}B_h + \tau\beta_{k-1}A_h, \ldots, \alpha_0 B_h + \tau\beta_0 A_h,\ 0,\ 0, \ldots. \end{aligned}$$

I is the $d \times d$ identity matrix. Their transforms are given, respectively, by

$$I \qquad \text{and} \qquad z^{-k}\left(a(z)B_h + \tau b(z)A_h\right).$$

Now, consider the sequence

$$\underbrace{q_\tau \star q_\tau \star \cdots \star q_\tau}_{\nu_2 \text{ times}} \star \left(i_\tau - p\, w_\tau \;\star\; r\, v_\tau \right) \star \underbrace{q_\tau \star q_\tau \star \cdots \star q_\tau}_{\nu_1 \text{ times}} . \tag{5.10}$$

If conditions (5.8) and (5.9) are satisfied, it follows that this sequence is in l_1. (l_1 is closed under convolution and addition. The multiplication of an l_1-sequence by a matrix is an l_1-sequence.) The discrete Laplace transform of sequence (5.10) equals $\mathbf{M}_\tau(z)$, hence the sequence equals the kernel of $\mathcal{M}_\tau$. To conclude, $\mathcal{M}_\tau$ is bounded under conditions (5.8) and (5.9).

Suppose one of these conditions is violated. That is to say, there is a z with $|z| \geq 1$ such that $\det(a(z)M_{B_h} + \tau b(z)N_{B_h}) = 0$ or $\det(a(z)B_H + \tau b(z)A_H) = 0$. That would mean that $\frac{a}{b}(z) \in \sigma(-\tau M_{B_h}^{-1}M_{A_h}) \cup \sigma(-\tau B_H^{-1}A_H)$. Since $|z| \geq 1$ this violates the assumption of the lemma. □

Remark 5.2. The assumption of Lemma 5.2 implies the two-grid discrete solvability conditions (5.5).

Remark 5.3. The assumption of Lemma 5.2 implies that all poles of $\mathbf{M}(z)$ are inside the scaled stability region $\frac{1}{\tau}S$.

THEOREM 5.3. *Assume $\sigma(-\tau M_{B_h}^{-1}M_{A_h}) \cup \sigma(-\tau B_H^{-1}A_H) \subset \operatorname{int} S$, and consider $\mathcal{M}_\tau$ as an operator in $l_p(\infty)$, with $1 \leq p \leq \infty$. Then,*

$$\rho(\mathcal{M}_\tau) = \sup\{\rho(\mathbf{M}(z)) | \tau z \in \mathbb{C} \setminus \operatorname{int} S\} \tag{5.11}$$

$$= \sup_{\tau z \in \partial S} \rho(\mathbf{M}(z)). \tag{5.12}$$

Proof. The proof is a direct consequence of Lemma 2.2, and is similar to the proof of Theorem 4.4. □

Application of Lemma 2.3 yields the following result for the l_2-norm of $\mathcal{M}_\tau$.

THEOREM 5.4. *Assume $\sigma(-\tau M_{B_h}^{-1}M_{A_h}) \cup \sigma(-\tau B_H^{-1}A_H) \subset \operatorname{int} S$, and consider $\mathcal{M}_\tau$ as an operator in $l_2(\infty)$. Denote by $||\cdot||_2$ the l_2-norm* (2.2) *with $||\cdot||$ the standard Euclidean vector norm. Then,*

$$||\mathcal{M}_\tau||_2 = \sup\{||\mathbf{M}(z)|| : \tau z \in \mathbb{C} \setminus \mathit{int}\, S\} \tag{5.13}$$

$$= \sup_{\tau z \in \partial S} ||\mathbf{M}(z)||. \tag{5.14}$$

Remark 5.4. If the assumption of the former theorems is violated, but the weaker condition $\sigma(-\tau M_{B_h}^{-1}M_{A_h}) \cup \sigma(-\tau B_H^{-1}A_H) \subset \operatorname{int} S_{\gamma\tau}$ holds, then we can formulate a remark analogous to Remark 4.3.

5.3. Discrete-time versus continuous-time results. The relation between the two-grid operators $\mathcal{M}_\tau$ and $\mathcal{M}$ is similar to the relation between $\mathcal{K}_\tau$ and $\mathcal{K}$. More precisely, for both finite and infinite intervals,

$$\lim_{\tau \to 0} \rho(\mathcal{M}_\tau) = \rho(\mathcal{M}).$$

We also state the two-grid equivalent of Theorem 4.6, without proof.

THEOREM 5.5. *Assume $\sigma(-\tau M_{B_h}^{-1}M_{A_h}) \cup \sigma(-\tau B_H^{-1}A_H) \subset \Sigma_\alpha$. Consider $\mathcal{M}_\tau$ as an operator in $l_p(\infty)$ and $\mathcal{M}$ as an operator in $L_p(0, \infty)$, with $1 \leq p \leq \infty$. Then,*

TABLE 6.1
Theoretical and measured values of $\rho(\mathcal{K}_\tau)$ *for* (6.1) ($h = 1/16, \tau = 1/100$).

	CN	BDF(1)	BDF(2)	BDF(3)	BDF(4)	BDF(5)
finite length	0.458	0.658	0.548	0.486	0.445	0.414
infinite length	0.962	0.962	0.962	0.976	1.149	1.865
measured	0.960	0.961	0.961	0.974	1.147	1.858

TABLE 6.2
Theoretical and measured values of $\rho(\mathcal{M}_\tau)$ *for* (6.1) ($h = 1/16, \tau = 1/100$).

	CN	BDF(1)	BDF(2)	BDF(3)	BDF(4)	BDF(5)
finite length	0.050	0.050	0.052	0.051	0.049	0.047
infinite length	0.264	0.069	0.106	0.170	0.343	1.184
measured	0.255	0.064	0.099	0.161	0.335	1.166

(i) *if the linear multistep method is A-stable, then* $\rho(\mathcal{M}_\tau) \le \rho(\mathcal{M})$;

(ii) *if the linear multistep method is* $A(\alpha)$*-stable, then*

$$\rho(\mathcal{M}_\tau) \le \sup_{z\in\Sigma_\alpha^c} \rho(\mathbf{M}(z)) = \sup_{z\in\partial\Sigma_\alpha^c} \rho(\mathbf{M}(z)) \,, \tag{5.15}$$

with $\Sigma_\alpha^c = \mathbb{C} \setminus \Sigma_\alpha = \{z : |\mathrm{Arg}(z)| \le \pi - \alpha\}$.

6. Model problem analysis.

6.1. A one-dimensional model problem. In order to clarify the convergence behaviour of the waveform relaxation methods, we shall start with a very simple and small model problem, the one-dimensional heat equation on the unit interval,

$$\frac{\partial \mathbf{u}}{\partial t} - \frac{\partial^2 \mathbf{u}}{\partial x^2} = 0 \,, \quad x \in [0,1] \,. \tag{6.1}$$

Dirichlet boundary and initial conditions are chosen such that the solution equals $\mathbf{u}(x,t) = \sin(\pi x)\exp(-\pi^2 t)$. The problem is discretised using linear finite elements on a mesh Ω_h with mesh size $h = 1/16$.

We consider the Gauss–Seidel waveform relaxation algorithm and the two-level method, with one red/black Gauss–Seidel presmoothing step, a similar postsmoothing step, standard coarsening ($H = 2h$), and linear interpolation. The restriction is defined in the standard way for finite-element multigrid methods, i.e., $r = p^t$. For both waveform algorithms, we analyse the use of different time-discretisation formulae, with a constant time step $\tau = 1/100$. In particular, we consider the trapezoidal rule or Crank–Nicolson (CN) method, and the backward differentiation formulae (BDF) of order 1 up to 5. The spectral radii of the finite and infinite time-interval operators for the standard and for the two-level algorithm are reported in Tables 6.1 and 6.2, respectively. The results were computed by direct numerical evaluation of formulae (4.11) and (4.15), and (5.7) and (5.12). The tables also present values of convergence factors, observed with an implementation of the methods, using 1000 time steps. An oscillatory initial approximation to the solution was chosen in order to excite all possible error frequencies. The measured values correspond very well to the theoretical, infinite-interval spectral radii. This effect is explained in more detail in §7.1.

These results can be understood by looking at the *spectral picture* [21, p. 107], which facilitates a graphical inspection of convergence. In the spectral picture a set of contour lines of the function $\rho(\mathbf{K}(z))$ or $\rho(\mathbf{M}(z))$ is plotted for z in a region of the complex plane close to the complex origin. On top of this picture, the scaled stability boundary of the linear multistep methods can be plotted. Figures 6.1 and 6.2 display contour lines of $\rho(\mathbf{K}(z))$ and $\rho(\mathbf{M}(z))$ (for

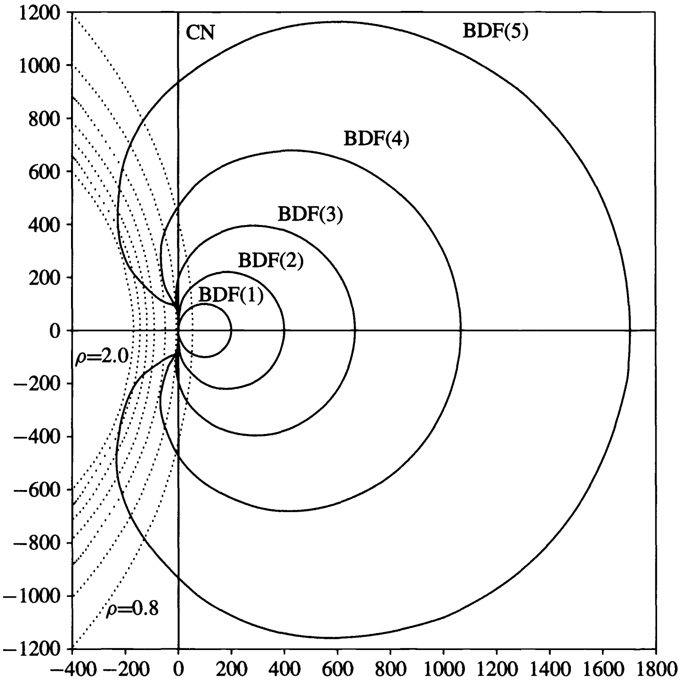


FIG. 6.1. *Spectral picture and graphical convergence test for* (6.1) ($\rho(\mathbf{K}(z))$, $h = 1/16$, $\tau = 1/100$).

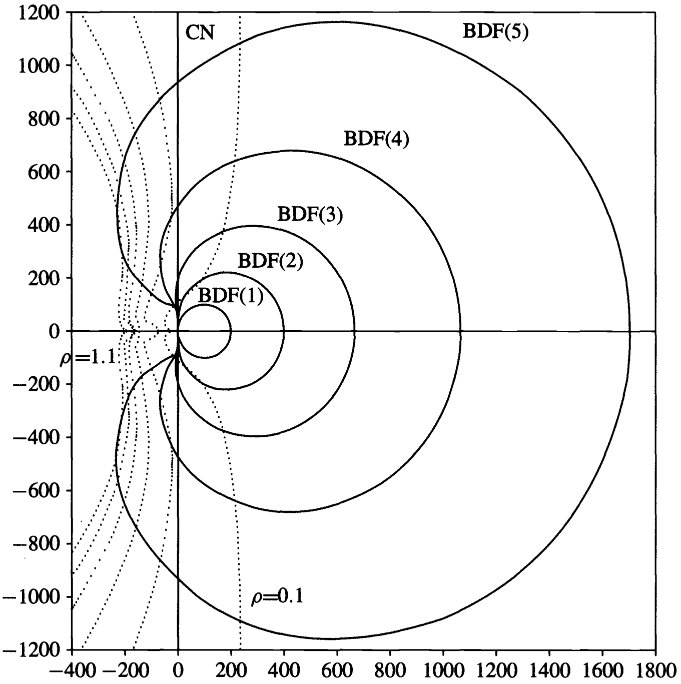


FIG. 6.2. *Spectral picture and graphical convergence test for* (6.1) ($\rho(\mathbf{M}(z))$, $h = 1/16$, $\tau = 1/100$).

values 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0 and for values 0.1, 0.3, 0.5, 0.7, 0.9, 1.1, respectively) for the model problem, together with the scaled stability region boundaries of the CN and BDF methods.

The values of the finite-interval spectral radii can be estimated by checking the values of the functions at the points on the real axis given by $\frac{1}{\tau}\frac{\alpha_k}{\beta_k}$ (which are not shown in the picture). With increasing order of the BDF methods, these points move to the right. Indeed, $\frac{\alpha_k}{\beta_k}$ equals 1 (BDF(1)), 3/2 (BDF(2)), 11/6 (BDF(3)), 25/12 (BDF(4)), and 137/60 (BDF(5)). A value of 2 is found for the CN method.

The values of the infinite-interval spectral radii can be estimated by taking the maximum of $\rho(\mathbf{K}(z))$ or $\rho(\mathbf{M}(z))$ over the plotted scaled stability region boundaries. The infinite-length discrete-time waveform methods are convergent for the CN method and the low-order BDF methods. Divergence is observed for some high-order methods. In general, the spectral radius increases with increasing order of the BDF method. This was to be expected from Theorems 4.6 and 5.5, and the knowledge that the BDF methods are $A(\alpha)$-stable with $\alpha = 90°$ (BDF(1), BDF(2)), $\alpha = 88°$ (BDF(3)), $\alpha = 73°$ (BDF(4)), and $\alpha = 51°$ (BDF(5)). Note also that the maximum of $\rho(\mathbf{K}(z))$ over $\frac{1}{\tau}\partial S$ is found at the origin for CN, BDF(1), and BDF(2). Hence, the equality of the corresponding values in Table 6.1.

6.2. A two-dimensional model problem. Next, we study the two-dimensional heat equation on the unit square,

$$\frac{\partial \mathbf{u}}{\partial t} - \frac{\partial^2 \mathbf{u}}{\partial x^2} - \frac{\partial^2 \mathbf{u}}{\partial y^2} = 0\,, \quad (x, y) \in [0,1] \times [0,1]\,, \tag{6.2}$$

completed with Dirichlet boundary conditions and an initial condition. The analytical solution equals $\mathbf{u}(x, y, t) = 1 + \sin(\pi x/2)\sin(\pi y/2)\exp(-\pi^2 t/2)$. The problem is discretised on a regular triangular mesh with linear elements, and on a regular rectangular mesh with bilinear elements. We will analyse convergence of the two-level method, with one four-colour Gauss–Seidel presmoothing step, a similar postsmoothing step, standard coarsening ($H = 2h$), and linear interpolation. The restriction is again defined by $r = p^t$, which leads to a seven-point formula in the linear element case, and a nine-point formula in the bilinear element case.

It is no longer practical to use a direct numerical evaluation of $\rho(\mathbf{M}(z))$ to study convergence characteristics. Instead, we can resort to Remark 5.1, which relates $\rho(\mathbf{M}(z))$ to the analysis of a standard two-grid method for a simple elliptic problem. The latter can be analysed efficiently using a classical Fourier mode analysis as introduced by Brandt in [1]. Fourier analysis shows that, under certain conditions, matrix $\mathbf{M}(z)$ is spectrally equivalent to a block-diagonal matrix whose diagonal blocks are matrices of size at most four by four. The general form of these four-by-four matrices can be derived by studying the action of the different multigrid operators on certain sets of four related exponential or sinusoidal Fourier modes. The spectral properties of $\mathbf{M}(z)$ are then calculated easily. We refer to the above reference, and to [19] and [24] for an in-depth discussion of the classical Fourier mode analysis. In the present paper we have closely followed the guidelines laid out in [24, Chap. 7].

Figure 6.3 shows the spectral picture for linear finite elements with h=1/32. In the computation we used exponential Fourier modes. They led to an exact value of the spectral radius in the case of periodic boundary conditions. A slight modification to the standard exponential mode analysis was applied to cater for the Dirichlet boundary conditions, a modification described in [24, p. 111]. Figure 6.4 shows a similar picture for bilinear elements on a grid with h=1/32. The nature of the stencil in the bilinear element case is such that a sinusoidal Fourier mode analysis is possible, see [19, §7.1]. The sinusoidal mode analysis leads automatically to the correct value of the spectral radius in the case of a problem with Dirichlet boundary conditions.

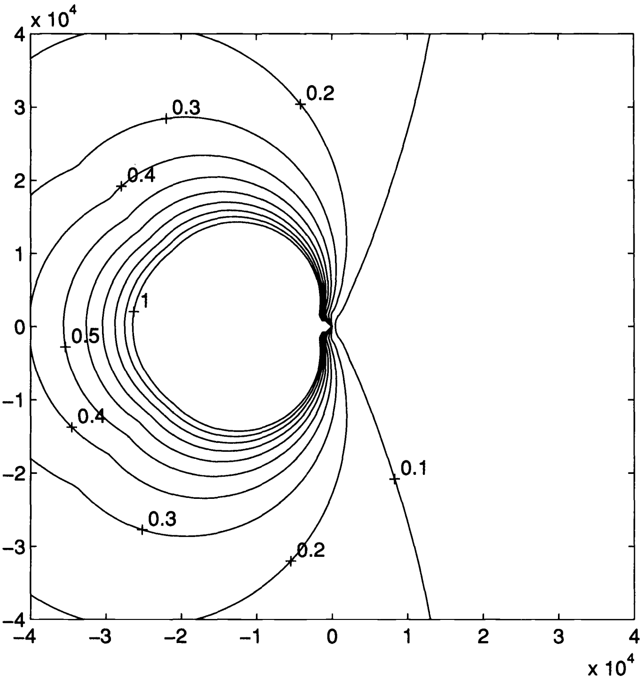


FIG. 6.3. *Spectral picture for* (6.2) ($\rho(\mathbf{M}(z))$, *linear elements*, $h = 1/32$).

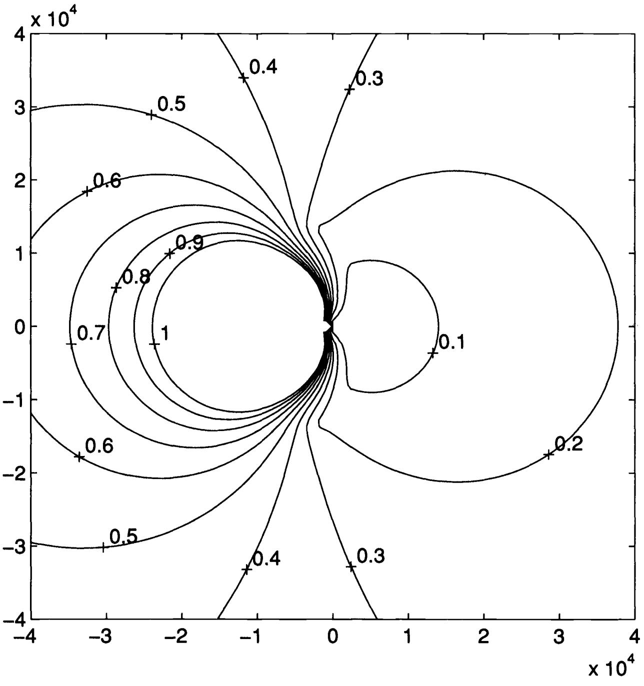


FIG. 6.4. *Spectral picture for* (6.2) ($\rho(\mathbf{M}(z))$, *bilinear elements*, $h = 1/32$).

TABLE 6.3
Theoretical and measured values of $\rho(\mathcal{M}_\tau)$ *for* (6.2) (*linear elements,* $h = 1/32$, $\tau = 1/100$).

	CN	BDF(1)	BDF(2)	BDF(3)	BDF(4)	BDF(5)
finite length	0.102	0.120	0.110	0.104	0.100	0.097
infinite length	0.374	0.148	0.148	0.150	0.170	0.233
measured	0.329	0.135	0.137	0.138	0.150	0.198

TABLE 6.4
Theoretical and measured values of $\rho(\mathcal{M}_\tau)$ *for* (6.2) (*bilinear elements,* $h = 1/32$, $\tau = 1/100$).

	CN	BDF(1)	BDF(2)	BDF(3)	BDF(4)	BDF(5)
finite length	0.038	0.042	0.040	0.038	0.037	0.036
infinite length	0.356	0.052	0.058	0.068	0.087	0.132
measured	0.313	0.039	0.044	0.049	0.067	0.118

In Tables 6.3 and 6.4 we present two-grid spectral radii for the finite-length and infinite-length waveform operators, together with two-grid convergence factors computed numerically using an oscillatory initial approximation and 1000 time steps. As for the one-dimensional problem, there is again a good agreement between the theoretical infinite-length spectral radii and the experimental convergence factors.

7. Numerical experiments.

7.1. The one-dimensional model problem.

In this section, we shall clarify the relation between the finite and infinite time-interval spectral radii. To this end, we solve (6.1) using the Gauss–Seidel waveform method with BDF(2) and BDF(5) time discretisation, with constant time step $\tau = 1/100$ on 100 time levels ($N = 100$). (Note that a similar analysis could be done for the two-level method and/or for larger values of N. It would lead to similar conclusions and insights.)

Let $d_\tau^{(\nu)}$ denote the discrete defect or residual in the νth iteration. The convergence factor of the νth iteration is then defined by

$$\rho^{(\nu)} = ||d_\tau^{(\nu)}||_2/||d_\tau^{(\nu-1)}||_2 . \tag{7.1}$$

In Fig. 7.1 successive convergence factors are plotted for the first 400 waveform Gauss–Seidel iterations, when BDF(2) discretisation is used. These factors appear to remain more or less constant for a large number of iterations. The height of the plateau matches the value obtained in Table 6.1 for infinite time intervals, i.e., 0.962. Eventually, the plateau in Fig. 7.1 is left, and the factors start to decrease. Ultimately, they start to rise again and reach the value 1. This is for purely technical reasons, because at that time the solution has converged within the finite-precision arithmetic of the implementation. A similar plot is given in Fig. 7.2 for the BDF(5) discretisation. Here, the evolution is much more erratic. The results clearly indicate divergence for a large number of iterations. After a sufficient number of iterations, the convergence factors decrease below 1, and the iteration starts to converge rapidly.

This behaviour can be explained by examining the *time-level convergence factors*. These factors are similar to the standard convergence factors (7.1), but are evaluated for each time level separately,

$$\rho_k^{(\nu)} = ||d_k^{(\nu)}||_2/||d_k^{(\nu-1)}||_2 .$$

In Fig. 7.3, we plotted such time-level convergence factors for the BDF(2) method (for $\nu = 10$, $\nu = 100$, $\nu = 200$, and $\nu = 300$). The factor measured at the first time level equals

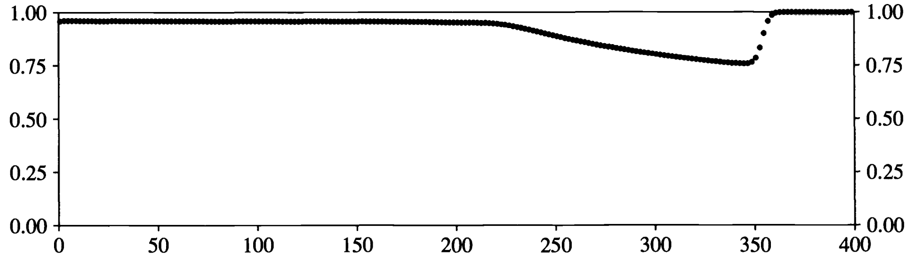


FIG. 7.1. *Convergence factors* $\rho^{(\nu)}$ *as a function of* ν *(BDF(2) method).*

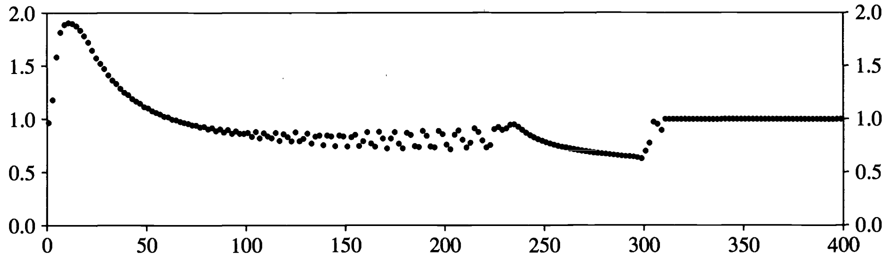


FIG. 7.2. *Convergence factors* $\rho^{(\nu)}$ *as a function of* ν *(BDF(5) method).*

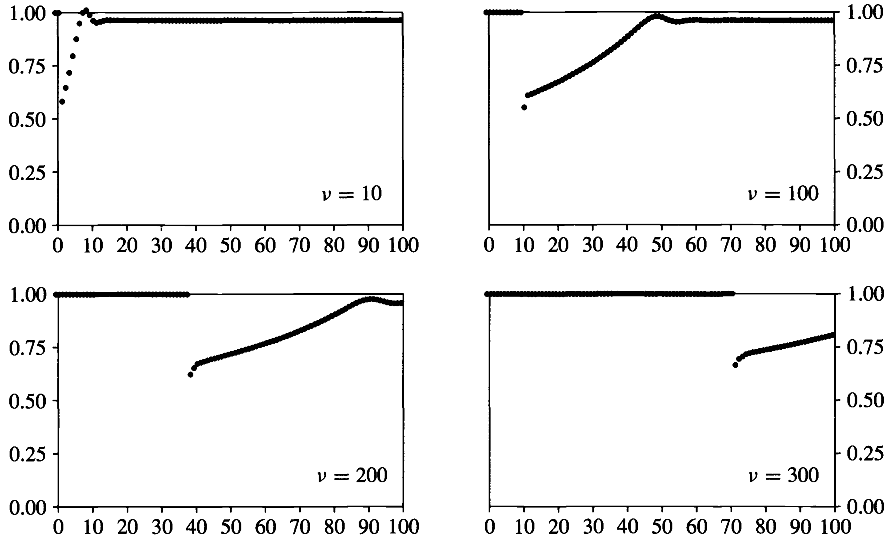


FIG. 7.3. *Time-level convergence factors* $\rho_k^{(\nu)}$ *as a function of k (BDF(2) method).*

0.548, exactly equal to the value predicted by the finite time-interval analysis in Table 6.1. The convergence factors at the next time levels increase, and eventually become constant. The height of the plateau matches the spectral radius value for infinite time intervals. As more iterations are applied, the plateau is forced out of the time window and the corresponding convergence factors decrease.

In Fig. 7.4, we have plotted time-level convergence factors for the BDF(5) method (for $\nu = 1$, $\nu = 5$, $\nu = 50$, and $\nu = 100$). Again, we observe that the factor at the first time level corresponds to the value predicted by the finite time-interval analysis (0.414). The pictures illustrate the onset of oscillations which rapidly explode. As more iterations are applied, the

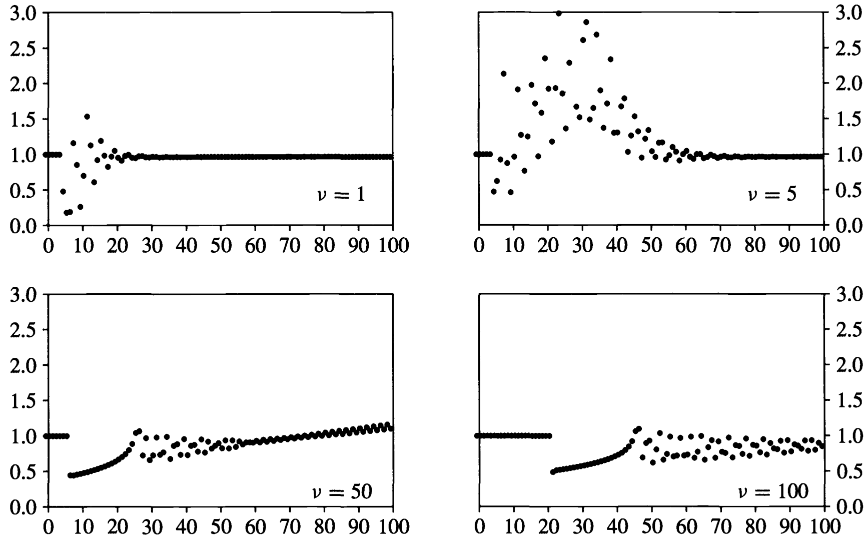


FIG. 7.4. *Time-level convergence factors* $\rho_k^{(\nu)}$ *as a function of k (BDF(5) method).*

TABLE 7.1
Averaged convergence factors for (6.2) (*linear basis functions*, $h = 1/32$, $\tau = 1/200$).

	CN	BDF(1)	BDF(2)	BDF(3)	BDF(4)	BDF(5)
Gauss–Seidel	0.990	0.990	0.990	0.997	–	–
V-cycle	0.438	0.177	0.177	0.275	0.844	–
W-cycle	0.307	0.124	0.124	0.124	0.381	–

TABLE 7.2
Averaged convergence factors for (6.2) (*bilinear basis functions*, $h = 1/32$, $\tau = 1/200$).

	CN	BDF(1)	BDF(2)	BDF(3)	BDF(4)	BDF(5)
Gauss–Seidel	0.985	0.985	0.985	0.996	–	–
V-cycle	0.446	0.046	0.132	0.332	–	–
W-cycle	0.295	0.041	0.041	0.052	0.538	–

region of divergent behaviour moves to the right, and is forced out of the time window. From then on, the iteration converges rapidly.

7.2. The two-dimensional problem. We discretise (6.2) using linear basis functions (triangular elements) or bilinear basis functions (rectangular elements) on a mesh with equal mesh size in x- and y-direction. The resulting system of ODEs is solved using Gauss–Seidel and multigrid waveform relaxation for $t \in [0, 1]$. In the latter we applied V-cycles and W-cycles, with one presmoothing and one postsmoothing step of four-colour Gauss–Seidel type. We use standard coarsening down to a mesh with size $h = 1/2$, seven-point prolongation (linear basis functions), and nine-point prolongation (bilinear basis functions). The restriction operator is defined as $r = p^t$.

In Tables 7.1 and 7.2 we report *averaged convergence factors.* These are defined as the average of $\rho^{(\nu)}$ over the region of nearly constant behaviour. The dashes ("–") in the tables indicate that the corresponding method showed divergence over a large number of iterations. Both tables illustrate the dependence of the convergence on the nature of the time-discretisation method.

In Tables 7.3–7.8, we report averaged convergence factors obtained with W-cycles for different values of the mesh-size parameters, and for different discretisation schemes. We observe a dependence of the actual convergence factors on h and τ. For the Crank–Nicolson and BDF(2) methods, these factors appear to be bounded by a constant, smaller than one, independent of the mesh size.

For a constant value of h, we expect the convergence factors to converge to the continuous-time results when τ decreases, see §§4.3 and 5.3. This behaviour is recognised clearly for the CN method in Tables 7.3 and 7.6. Due to the shape of the stability regions of the BDF(2) and BDF(4) methods, it takes a much smaller value of τ before the discrete-time convergence factors tend to the continuous-time ones, see Tables 7.4, 7.5, 7.7, and 7.8. For a constant value of τ, we observe an initial increase of the convergence factor when h decreases. For sufficiently small h the convergence factor starts to decrease again. This behaviour is similar to what is observed when the multigrid waveform relaxation method is used to solve the ODEs obtained by spatial finite-difference discretisation of a parabolic problem. We refer to [21, §3.5] for an intuitive explanation, and to [22] for a discussion based on an exponential Fourier mode analysis.

Appendix. Analysis of the multigrid waveform relaxation operators. We consider the case where we have a hierarchy of grids, $\Omega_{h_0} \subset \Omega_{h_1} \subset \cdots \subset \Omega_{h_l}$, a set of prolongation operators $p_{h_i}^{h_{i+1}} : \Omega_{h_i} \longrightarrow \Omega_{h_{i+1}}$, $0 \le i \le l-1$, a set of restriction operators, $r_{h_{i+1}}^{h_i} : \Omega_{h_{i+1}} \longrightarrow \Omega_{h_i}$, $0 \le i \le l-1$, and discretisation matrices B_{h_i} and A_{h_i}, $0 \le i \le l$. The multigrid algorithm differs from the two-grid cycle in that the coarse-grid defect equation is approximately solved by an application of γ two-grid cycles, an idea that is further extended recursively. (The classical V- and W-cycles correspond to $\gamma = 1$ and $\gamma = 2$, respectively.) In the continuous-time case this leads to an iteration of the form $u^{(\nu)} = \mathcal{M}_{h_l} u^{(\nu-1)} + \varphi$. In the discrete-time case we end up with an iteration operator which we denote by $(\mathcal{M}_{h_l})_\tau$.

Both iterative schemes can be analysed in exactly the same way as the two-grid cycles have been analysed. A Laplace-transform argument is used in the continuous-time case, whereas the discrete-time case is treated by using a discrete Laplace-transform method. Proceeding as before, we can derive the symbol of the continuous-time multigrid waveform relaxation method, $\mathbf{M}_{\mathbf{h_l}}(z)$. The latter takes a particularly simple form under the natural assumption that the semidiscretised PDE operators are invertible. In that case we can apply the following Lemma.

LEMMA A.1. *Let $B\dot{u} + Au = f$ have a unique solution, and let it be solved approximately by γ steps of a consistent waveform method: $u^{(k)} = \mathcal{M}u^{(k-1)} + \varphi$ with $u^{(0)}(t) = 0$. Then, the γth iterate can be represented as $u^{(\gamma)} = (I - \mathcal{M}^\gamma)u$.*

Under the above assumption, the multigrid symbol becomes

$$\mathbf{M}_{\mathbf{h_l}}(z) = \begin{cases} \mathbf{K}_{\mathbf{h_l}}^{\nu_2}(z)\left(I - p_{h_{l-1}}^{h_l}\left(I - \mathbf{M}_{\mathbf{h_{l-1}}}^{\gamma}(z)\right)\mathbf{L}_{\mathbf{h_{l-1}}}^{-1}(z) r_{h_l}^{h_{l-1}} \mathbf{L}_{\mathbf{h_l}}(z)\right)\mathbf{K}_{\mathbf{h_l}}^{\nu_1}(z)\,, & \\ & l \neq 1, \\ \mathbf{K}_{\mathbf{h_1}}^{\nu_2}(z)\left(I - p_{h_0}^{h_1}\mathbf{L}_{\mathbf{h_0}}^{-1}(z) r_{h_1}^{h_0}\mathbf{L}_{\mathbf{h_1}}(z)\right)\mathbf{K}_{\mathbf{h_1}}^{\nu_1}(z)\,, & l = 1, \end{cases}$$

where $\mathbf{L}_{\mathbf{h_i}}(z) = zB_{h_i} + A_{h_i}$ and $\mathbf{K}_{\mathbf{h_i}}(z) = (zM_{B_{h_i}} + M_{A_{h_i}})^{-1}(zN_{B_{h_i}} + N_{A_{h_i}})$. Note that $\mathbf{M}_{\mathbf{h_l}}(z)$ is technically more complicated when the assumption is violated. In that case, it does not involve the factor $\mathbf{L}_{\mathbf{h_{l-1}}}^{-1}(z)$.

Remark A.1. Let $\mathbf{K}_{\mathbf{h_i}}(z)$ correspond to a Gauss–Seidel or weighted Jacobi splitting. Then, $\mathbf{M}_{\mathbf{h_l}}(z)$ is the multigrid iteration operator for the elliptic problem $(zB_{h_l} + A_{h_l})u_{h_l} = f_{h_l}$; compare [4, p. 162] and [19, p. 46].

TABLE 7.3
Averaged convergence factors for (6.2) (*linear basis functions, CN method, W-cycle*).

h, τ	0.04	0.02	0.01	0.005	0.0025	0.001
1/4	0.103	0.135	0.134	0.135	0.134	0.135
1/8	0.126	0.256	0.305	0.304	0.304	0.304
1/16	0.117	0.135	0.282	0.359	0.358	0.357
1/32	0.123	0.125	0.140	0.307	0.372	0.371

TABLE 7.4
Averaged convergence factors for (6.2) (*linear basis functions, BDF*(2) *method, W-cycle*).

h, τ	0.04	0.02	0.01	0.005	0.0025	0.001
1/4	0.051	0.070	0.111	0.128	0.133	0.134
1/8	0.086	0.086	0.108	0.194	0.247	0.291
1/16	0.118	0.118	0.118	0.118	0.124	0.266
1/32	0.124	0.124	0.124	0.124	0.124	0.125

TABLE 7.5
Averaged convergence factors for (6.2) (*linear basis functions, BDF*(4) *method, W-cycle*).

h, τ	0.04	0.02	0.01	0.005	0.0025	0.001
1/4	0.173	0.320	0.290	0.171	0.135	0.134
1/8	0.141	0.324	0.525	0.766	0.666	0.311
1/16	0.121	0.154	0.358	0.653	0.807	0.948
1/32	0.124	0.124	0.124	0.381	0.726	1.091

TABLE 7.6
Averaged convergence factors for (6.2) (*bilinear basis functions, CN method, W-cycle*).

h, τ	0.04	0.02	0.01	0.005	0.0025	0.001
1/4	0.102	0.133	0.136	0.137	0.137	0.137
1/8	0.150	0.231	0.285	0.293	0.294	0.294
1/16	0.080	0.179	0.268	0.330	0.343	0.344
1/32	0.042	0.086	0.184	0.295	0.343	0.355

TABLE 7.7
Averaged convergence factors for (6.2) (*bilinear basis functions, BDF*(2) *method, W-cycle*).

h, τ	0.04	0.02	0.01	0.005	0.0025	0.001
1/4	0.049	0.072	0.085	0.088	0.106	0.125
1/8	0.063	0.088	0.130	0.178	0.224	0.241
1/16	0.045	0.046	0.047	0.104	0.167	0.246
1/32	0.041	0.041	0.041	0.041	0.042	0.132

TABLE 7.8
Averaged convergence factors for (6.2) (*bilinear basis functions, BDF*(4) *method, W-cycle*).

h, τ	0.04	0.02	0.01	0.005	0.0025	0.001
1/4	0.124	0.217	0.161	0.148	0.139	0.135
1/8	0.158	0.319	0.600	0.661	0.405	0.324
1/16	0.069	0.147	0.377	0.735	0.892	0.770
1/32	0.042	0.048	0.112	0.538	0.646	0.937

As in [10], the continuous-time convergence theorems can be formulated in terms of this symbol. The ideas behind the proofs are identical to the ones behind the corresponding proofs in the above reference, and therefore omitted.

THEOREM A.2. *The multigrid waveform relaxation operator $\mathcal{M}_{h_l}$ is a bounded operator in $C[0, T]$ and*

$$\rho(\mathcal{M}_{h_l}) = \rho\left(\mathbf{M}_{\mathbf{h_l}}(\infty)\right) . \tag{A.2}$$

THEOREM A.3. *Assume that all eigenvalues of $M_{B_{h_i}}^{-1} M_{A_{h_i}}$, $1 \le i \le l$, and $B_{h_0}^{-1} A_{h_0}$ have positive real parts, and consider $\mathcal{M}_{h_l}$ as an operator in $L_p(0, \infty)$ with $1 \le p \le \infty$. Then, $\mathcal{M}_{h_l}$ is a bounded operator with spectral radius*

$$\rho(\mathcal{M}_{h_l}) = \sup_{\mathrm{Re}(z) \ge 0} \rho\left(\mathbf{M}_{\mathbf{h_l}}(z)\right) = \sup_{\xi \in \mathbb{R}} \rho\left(\mathbf{M}_{\mathbf{h_l}}(i\xi)\right) . \tag{A.3}$$

Following the line of arguments of §5.1, we can derive the discrete-time symbol

$$(\mathbf{M}_{\mathbf{h_l}})_\tau(z) = \mathbf{M}_{\mathbf{h_l}}\left(\frac{1}{\tau}\frac{a}{b}(z)\right) .$$

The discrete-time convergence theorems are immediate extensions of Theorems 5.1 and 5.3. The proofs are very similar.

THEOREM A.4. *Assume $\frac{\alpha_k}{\beta_k} \notin \bigcup_{i=1}^{l} \sigma(-\tau M_{B_{h_i}}^{-1} M_{A_{h_i}}) \cup \sigma\left(-\tau B_{h_0}^{-1} A_{h_0}\right)$, and consider $(\mathcal{M}_{h_l})_\tau$ as an operator in $l_p(N)$, with $1 \le p \le \infty$ and N finite. Then, $(\mathcal{M}_{h_l})_\tau$ is a bounded operator and*

$$\rho((\mathcal{M}_{h_l})_\tau) = \rho\left(\mathbf{M}_{\mathbf{h_l}}\left(\frac{1}{\tau}\frac{\alpha_k}{\beta_k}\right)\right) . \tag{A.4}$$

THEOREM A.5. *Assume $\bigcup_{i=1}^{l} \sigma(-\tau M_{B_{h_i}}^{-1} M_{A_{h_i}}) \cup \sigma(-\tau B_{h_0}^{-1} A_{h_0}) \subset \mathrm{int}\, S$, and consider $(\mathcal{M}_{h_l})_\tau$ as an operator in $l_p(\infty)$, with $1 \le p \le \infty$. Then,*

$$\rho((\mathcal{M}_{h_l})_\tau) = \sup\{\rho(\mathbf{M}_{\mathbf{h_l}}(z)) | \tau z \in \mathbb{C} \setminus \mathrm{int}\, S\} = \sup_{\tau z \in \partial S} \rho(\mathbf{M}_{\mathbf{h_l}}(z)) . \tag{A.5}$$

REFERENCES

[1] A. BRANDT, *Multi-level adaptive solutions to boundary-value problems*, Math. Comp., 31 (1977), pp. 333–390.

[2] P. BRENNER, V. THOMÉE, AND L. B. WAHLBIN, *Besov Spaces and Applications to Difference Methods for Initial Value Problems*, Lecture Notes in Mathematics 434, Springer-Verlag, Berlin, 1975.

[3] W. HACKBUSCH, *Parabolic multi-grid methods*, in Computing Methods in Applied Sciences and Engineering VI, R. Glowinski and J.-L. Lions, eds., North-Holland, Amsterdam, 1984, pp. 189–197.

[4] ———, *Multi-grid methods and applications*, Springer Series in Computational Mathematics 4, Springer-Verlag, Berlin, 1985.

[5] E. HAIRER AND G. WANNER, *Solving ordinary differential equations* II, Springer Series in Computational Mathematics 14, Springer-Verlag, Berlin, 1991.

[6] G. H. HARDY, J. E. LITTLEWOOD, AND G. PÓLYA, *Inequalities*, 2nd ed., Cambridge University Press, Cambridge, 1978.

[7] G. HORTON, *The time-parallel multigrid method*, Comm. Appl. Numer. Methods, 8 (1992), pp. 585–595.

[8] G. HORTON AND S. VANDEWALLE, *A space-time multigrid method for parabolic Partial Differential Equations*, SIAM J. Sci. Comput., 16 (1995), pp. 848–864.

[9] G. HORTON, S. VANDEWALLE, AND P. WORLEY, *An algorithm with polylog parallel complexity for solving parabolic partial differential equations*, SIAM J. Sci. Comput., 16 (1995), pp. 531–541.

[10] J. Janssen and S. Vandewalle, *Multigrid waveform relaxation on spatial finite element meshes: The continuous-time case*, SIAM J. Numer. Anal., 33 (1996).

[11] J. D. Lambert, *Computational Methods in Ordinary Differential Equations*, John Wiley and Sons, Chichester, 1973.

[12] E. Lelarasmee, A. E. Ruehli, and A. L. Sangiovanni-Vincentelli, *The waveform relaxation method for time-domain analysis of large scale integrated circuits*, IEEE Trans. CAD, 1 (1982), pp. 131–145.

[13] C. Lubich, *On the stability of linear multistep methods for Volterra convolution equations*, IMA J. Numer. Anal., 3 (1983), pp. 439–465.

[14] C. Lubich and A. Ostermann, *Multi-grid dynamic iteration for parabolic equations*, BIT, 27 (1987), pp. 216–234.

[15] U. Miekkala and O. Nevanlinna, *Convergence of dynamic iteration methods for initial value problems*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 459–482.

[16] ———, *Sets of convergence and stability regions*, BIT, 27 (1987), pp. 554–584.

[17] O. Nevanlinna, *Remarks on Picard-Lindelöf iteration*, II, BIT, 29 (1989), pp. 535–562.

[18] L. Reichel and L. N. Trefethen, *Eigenvalues and pseudo-eigenvalues of Toeplitz matrices*, Linear Algebra Appl., 162–164 (1992), pp. 153–185.

[19] K. Stüben and U. Trottenberg, *Multigrid methods: Fundamental algorithms, model problem analysis and applications*, in Multigrid Methods, U. Trottenberg and W. Hackbusch, eds., Lecture Notes in Mathematics 960, Springer-Verlag, Berlin, 1982, pp. 1–176.

[20] V. Thomée, *Galerkin finite element methods for parabolic problems*, Lecture Notes in Mathematics 1054, Springer-Verlag, Berlin, 1984.

[21] S. Vandewalle, *Parallel Multigrid Waveform Relaxation for Parabolic Problems*, Teubner, Stuttgart, 1993.

[22] S. Vandewalle and G. Horton, *Fourier mode analysis of the multigrid waveform relaxation and time-parallel multigrid methods*, Computing, 54(4), (1995), pp. 317–330.

[23] S. Vandewalle and E. Van de Velde, *Space-time concurrent multigrid waveform relaxation*, Annals of Numerical Mathematics, 1 (1994), pp. 347–363.

[24] P. Wesseling, *An Introduction to Multigrid Methods*, John Wiley and Sons, Chichester, 1992.

[25] K. Yosida, *Functional Analysis*, Springer-Verlag, New York, 1980.

SIAM J. SCI. COMPUT.
Vol. 17, No. 1, pp. 156–179, January 1996

011

IMPLICIT EXTRAPOLATION METHODS FOR MULTILEVEL FINITE ELEMENT COMPUTATIONS*

MICHAEL JUNG† AND ULRICH RÜDE‡

Abstract. Extrapolation methods for the solution of partial differential equations are commonly based on the existence of error expansions for the approximate solution. Implicit extrapolation, by contrast, is based on applying extrapolation indirectly, by using it on quantities like the residual. In the context of multigrid methods, a special technique of this type is known as τ-extrapolation. For finite element systems this algorithm can be shown to be equivalent to higher order finite elements. The analysis is local and does not use global expansions, so that the implicit extrapolation technique may be used on unstructured meshes and in cases where the solution fails to be globally smooth. Furthermore, the natural multilevel structure can be used to construct efficient multigrid and multilevel preconditioning techniques. The effectivity of the method is demonstrated for heat conduction problems and problems from elasticity theory.

Key words. finite elements, extrapolation, multigrid, elasticity

AMS subject classifications. 65F10, 65F50, 65N22, 65N50, 65N55

1. Introduction.

Multigrid methods have been shown to be very efficient solvers for elliptic partial differential equations (PDE). In this paper we are concerned with the so-called τ-extrapolation multigrid method (see Brandt [6] and Hackbusch [7]), which is an extension of conventional multigrid that can improve the accuracy of the numerical result by implicitly using higher order approximations.

In contrast to conventional extrapolation methods for partial differential equations, as described in Marchuk and Shaidurov [15] and Blum, Lin, and Rannacher [4], the τ-extrapolation algorithm is based on an *implicit* application of Richardson's deferred approach to the limit. We do not take linear combinations of computed approximations, but extrapolate the *residuals* of different levels. This is equivalent to forming a linear combination of the stiffness matrices and right-hand sides. The precise meaning of this will be explained in detail later.

We show that one step of multigrid τ-extrapolation for piecewise linear C^0 finite element (FE) methods is equivalent to using quadratic elements. This can be derived as a consequence of asymptotic error expansions for the numerical integration of the FE stiffness matrices, as shown in Rüde [20]. Here we will follow a different approach and show that the quadratic stiffness matrix and the stiffness matrix which is implicitly constructed by τ-extrapolation for linear elements coincide. Therefore the system solved by τ-extrapolation is equivalent to using quadratic elements. Furthermore, we show the asymptotically optimal convergence of a multigrid solution of the extrapolated system.

Our experimental framework is the *Finite Element Multi-Grid Package* (FEMGP) (see Steidten and Jung [22]) developed at the Technische Universität Chemnitz-Zwickau for the solution of elliptic and parabolic problems arising in the computation of magnetic and thermomechanical fields. We focus on self-adjoint second-order linear elliptic partial differential equations, using the heat conduction equation and the equations of elasticity as typical model problems. The equivalence of τ-extrapolation to higher order finite elements justifies its use even for unstructured meshes as produced with FEMGP; see also the results on τ-extrapolation-based higher order adaptive methods by McCormick and Rüde [16].

*Received by the editors May 27, 1994; accepted for publication (in revised form) December 9, 1994.

†Fakultät für Mathematik, Technische Universität Chemnitz-Zwickau, D-09107 Chemnitz, Germany (dr.michael.jung@mathematik.tu-chemnitz.de).

‡Institut für Informatik, Technische Universität München, Arcisstr. 21, D-80290 München 1, Germany (ruede@informatik.tu-muenchen.de).

2. Finite element discretizations of the boundary value problem. We consider two-dimensional second-order elliptic boundary value problems:

$$\text{(1)} \qquad \text{Find } u \in V_0 \text{ such that } \quad a(u,v) = \langle F, v\rangle \quad \text{for all } v \in V_0,$$

with a symmetric, V_0-elliptic, and V_0-bounded bilinear form $a(.,.)$; $\langle .,.\rangle : V_0^* \times V_0 \to \mathbb{R}^1$ is the duality pairing, V_0^* denotes the space which is dual to V_0, and $F \in V_0^*$ is a linear and bounded functional on V_0. Later we will describe more precisely which bilinear forms we want to investigate.

Let us first describe some finite element discretizations of problem (1). The starting point of the discretization process is a coarse triangular mesh $\mathcal{T}_1$. Then we generate a sequence of nested triangular meshes $\mathcal{T}_k = \{\delta_k^{(r)},\ r \in \mathcal{J}_k\}$, $k = 1,2,\dots,l$, $\mathcal{J}_k = \{1,2,\dots,R_k\}$, where R_k denotes the number of triangles of the triangulation $\mathcal{T}_k$. We suppose that we obtain the triangulation $\mathcal{T}_l$ by dividing all triangles $\delta_{l-1}^{(r)}$, $r \in \mathcal{J}_{l-1}$, into four congruent subtriangles $\delta_l^{(r)}$. The nodes of the triangulations are numbered *hierarchically*, i.e., $P^{(1)}, P^{(2)}, \dots, P^{(\overline{N}_1)}, P^{(\overline{N}_1+1)}, \dots, P^{(\overline{N}_2)}, \dots, P^{(\overline{N}_{k-1}+1)}, \dots, P^{(\overline{N}_k)}, \dots, P^{(\overline{N}_{l-1}+1)}, \dots, P^{(\overline{N}_l)}$, where $P^{(\overline{N}_{k-1}+1)}, \dots, P^{(\overline{N}_k)}$ are the nodes of $\mathcal{T}_k$ that do not belong to $\mathcal{T}_{k-1}$ (but are naturally also nodes of $\mathcal{T}_{k+1}, \dots, \mathcal{T}_l$).

Corresponding to each triangulation $\mathcal{T}_k$, $k = 1,2,\dots,l-1$, we define the finite element subspaces $V_k \subset V_0$ as

$$\text{(2)} \qquad V_k = V_k^{L,N} = \operatorname{span}\{p_k^{(i)} \ : \ i = 1,2,\dots,N_k\},$$

where the trial functions $p_k^{(i)}$ are piecewise linear functions such that $p_k^{(i)}$ is linear in all triangles of $\mathcal{T}_k$, they are continuous, and they satisfy the relations $p_k^{(i)}(x_1^{(j)}, x_2^{(j)}) = 1$ for $i = j$, $p_k^{(i)}(x_1^{(j)}, x_2^{(j)}) = 0$ for $i \neq j$, $i,j = 1,2,\dots,N_k$. Here $(x_1^{(j)}, x_2^{(j)})$ denotes the coordinates of the node $P^{(j)}$, and N_k is the number of nodes belonging to $\Omega \cup \Gamma_N$, where Γ_N is the part of the boundary $\partial\Omega$ on which natural boundary conditions are given.

The finite element subspace corresponding to the finest triangulation $\mathcal{T}_l$ we define, only formally for now, by

$$\text{(3)} \qquad V_l = \operatorname{span}\{\tilde{p}_l^{(i)},\ i = 1,2,\dots,N_l\}.$$

For the specific choice of the functions $\tilde{p}_l^{(i)}$ we consider four possibilities. The first one is the usual *nodal basis*, i.e., we set $\tilde{p}_l^{(i)} = p_l^{(i)}$, where the functions $p_l^{(i)}$ are defined in the same way as the functions $p_k^{(i)}$, $k = 1,2,\dots,l-1$. Consequently, we obtain the FE subspace

$$\text{(4)} \qquad V_l = V_l^{L,N} = \operatorname{span}\{p_l^{(i)},\ i = 1,2,\dots,N_l\}.$$

As a second possibility we use the *two-level h-hierarchical basis*, i.e.,

$$\text{(5)} \quad V_l = V_l^{L,H} = \operatorname{span}\{p_{l-1}^{(i)},\ i = 1,\dots,N_{l-1}\} \cup \operatorname{span}\{p_l^{(i)},\ i = N_{l-1}+1,\dots,N_l\}.$$

Additionally, to these two approaches we also introduce FE subspaces spanned by piecewise quadratic functions $q_{l-1}^{(i)}$. These functions are polynomials of degree 2 in all triangles of $\mathcal{T}_{l-1}$, they are continuous, and they satisfy the relations $q_{l-1}^{(i)}(x_1^{(j)}, x_2^{(j)}) = 1$ for $i = j$, $q_{l-1}^{(i)}(x_1^{(j)}, x_2^{(j)}) = 0$ for $i \neq j$, $i,j = 1,2,\dots,N_l$. Using these functions we can define the usual *quadratic nodal basis* as

$$\text{(6)} \qquad V_l = V_l^{Q,N} = \operatorname{span}\{q_{l-1}^{(i)},\ i = 1,2,\dots,N_l\},$$

and the *two-level p-hierarchical basis* as

$$(7)\quad V_l = V_l^{Q,H} = \text{span}\{p_{l-1}^{(i)},\ i = 1, \dots, N_{l-1}\} \cup \text{span}\{q_{l-1}^{(i)},\ i = N_{l-1}+1, \dots, N_l\ \}.$$

The sequence of FE subspaces V_k, $k = 1, 2, \dots, l$, where V_l stands for $V_l^{L,N}$, $V_l^{L,H}$, $V_l^{Q,N}$, or $V_l^{Q,H}$, respectively, results in a sequence of finite element schemes:

$$(8)\qquad \text{Find } u_k \in V_k \text{ such that } \quad a(u_k, v_k) = \langle F, v_k \rangle \quad \text{for all } v_k \in V_k.$$

The determination of the unknown function u_k is equivalent to the solution of the system

$$(9)\qquad K_k^{L,N} \underline{u}_k = \underline{f}_k^{L,N}$$

of the algebraic finite element equations, where for $k = 1, 2, \dots, l-1$,

$$(10)\qquad \underline{u}_k = [u_k^{(i)}]_{i=1,2,\dots,N_k} \quad \leftrightarrow u_k = \sum_{i=1}^{N_k} u_k^{(i)} p_k^{(i)},$$

$$(11)\qquad K_k^{L,N} = [K_k^{L,N,(ij)}]_{i,j=1,2,\dots,N_k}\ , \quad K_k^{L,N,(ij)} = a(p_k^{(j)}, p_k^{(i)}), \quad \text{and}$$

$$(12)\qquad \underline{f}_k^{L,N} = [f_k^{L,N,(i)}]_{i=1,2,\dots,N_k} \quad , \quad f_k^{L,N,(i)} = \langle F, p_k^{(i)} \rangle.$$

For $k = l$ the stiffness matrix K_l and the load vector $\underline{f}_l$ are defined in the same way, and we set only the functions $\tilde{p}_l^{(i)}$ instead of the functions $p_l^{(i)}$. Depending on the concrete choice of the functions $\tilde{p}_l^{(i)}$ (see the possibilities (4)–(7)), we get the stiffness matrices $K_l = K_l^{L,N}$, $K_l^{L,H}$, $K_l^{Q,N}$, or $K_l^{Q,H}$ and the load vectors $\underline{f}_l = \underline{f}_l^{L,N}$, $\underline{f}_l^{L,H}$, $\underline{f}_l^{Q,N}$, or $\underline{f}_l^{Q,H}$ respectively.

Next we specify the bilinear form $a(.,.)$. In the following we will consider bilinear forms which are defined by

$$(13)\qquad a(u, v) = \int_\Omega (A\nabla_x u, \nabla_x v)\, dx,$$

where A is a symmetric, positive definite (2×2)-matrix,

$$(14)\qquad \nabla_x = \left(\frac{\partial}{\partial x_1} \ \ \frac{\partial}{\partial x_2} \right)^T,$$

and $(.,.)$ denotes the Euclidian scalar product in the space $\mathbb{R}^2$. Such bilinear forms arise from the derivation of the weak formulation of heat conduction problems. Let us suppose that the entries of the matrix A are piecewise constant functions, i.e., constant in each triangle $\delta_{l-1}^{(r)}$, $r \in \mathcal{J}_{l-1}$. In this paper we will not discuss the variable coefficient case.

Next we prove an interesting relation among the matrices $K_{l-1}^{L,N}$, $K_l^{L,H}$, and $K_l^{Q,H}$, which is useful for the investigation of the convergence properties of a multigrid algorithm with extrapolation.

LEMMA 2.1. *Let $K_{l-1}^{L,N}$, $K_l^{L,H}$, and $K_l^{Q,H}$ be defined by the bilinear form* (13) *as described above, where the entries of the matrix $A(x)$ in the bilinear form are constant in each triangle $\delta_{l-1}^{(r)} \in \mathcal{T}_{l-1}$. Then the relation*

$$(15)\qquad K_l^{Q,H} = \frac{4}{3} K_l^{L,H} - \frac{1}{3} \tilde{K}_{l-1}$$

holds, where

$$\tilde{K}_{l-1} = \begin{pmatrix} K_{l-1}^{L,N} & 0 \\ 0 & 0 \end{pmatrix}.$$

Proof. Recall the definition of the stiffness matrices

$$K_{l-1}^{L,N} = [K_{l-1}^{L,N,(ij)}]_{i,j=1,2,\dots,N_{l-1}}, \quad K_{l-1}^{L,N,(ij)} = a(p_{l-1}^{(j)}, p_{l-1}^{(i)}), \tag{16}$$

$$K_l^{L,H} = [K_l^{L,H,(ij)}]_{i,j=1,2,\dots,N_l}, \tag{17}$$

$$K_l^{L,H,(ij)} = \begin{cases} a(p_{l-1}^{(j)}, p_{l-1}^{(i)}) & \text{for} \quad i,j = 1,2,\dots,N_{l-1}, \\ a(p_l^{(j)}, p_{l-1}^{(i)}) & \text{for} \quad j = N_{l-1}+1, N_{l-1}+2, \dots, N_l,\ i = 1,2,\dots,N_{l-1}, \\ a(p_{l-1}^{(j)}, p_l^{(i)}) & \text{for} \quad j = 1,2,\dots,N_{l-1},\ i = N_{l-1}+1, N_{l-1}+2, \dots, N_l, \\ a(p_l^{(j)}, p_l^{(i)}) & \text{for} \quad i,j = N_{l-1}+1, N_{l-1}+2, \dots, N_l\,, \end{cases}$$

$$K_l^{Q,H} = [K_l^{Q,H,(ij)}]_{i,j=1,2,\dots,N_l}, \tag{18}$$

$$K_l^{Q,H,(ij)} = \begin{cases} a(p_{l-1}^{(j)}, p_{l-1}^{(i)}) & \text{for} \quad i,j = 1,2,\dots,N_{l-1}, \\ a(q_{l-1}^{(j)}, p_{l-1}^{(i)}) & \text{for} \quad j = N_{l-1}+1, N_{l-1}+2, \dots, N_l,\ i = 1,2,\dots,N_{l-1}, \\ a(p_{l-1}^{(j)}, q_{l-1}^{(i)}) & \text{for} \quad j = 1,2,\dots,N_{l-1},\ i = N_{l-1}+1, N_{l-1}+2, \dots, N_l, \\ a(q_{l-1}^{(j)}, q_{l-1}^{(i)}) & \text{for} \quad i,j = N_{l-1}+1, N_{l-1}+2, \dots, N_l. \end{cases}$$

All these stiffness matrices have the structure

$$K_l = \begin{pmatrix} K_{l,vv} & K_{l,vm} \\ K_{l,mv} & K_{l,mm} \end{pmatrix}, \tag{19}$$

where $K_{l,vv}$ corresponds to the nodes of the triangulation $\mathcal{T}_{l-1}$, $K_{l,mm}$ corresponds to the new nodes in the triangulation $\mathcal{T}_l$, and $K_{l,mv}$, $K_{l,vm}$ are the coupling blocks.

From the definitions (16)–(18) of the matrix elements we see that

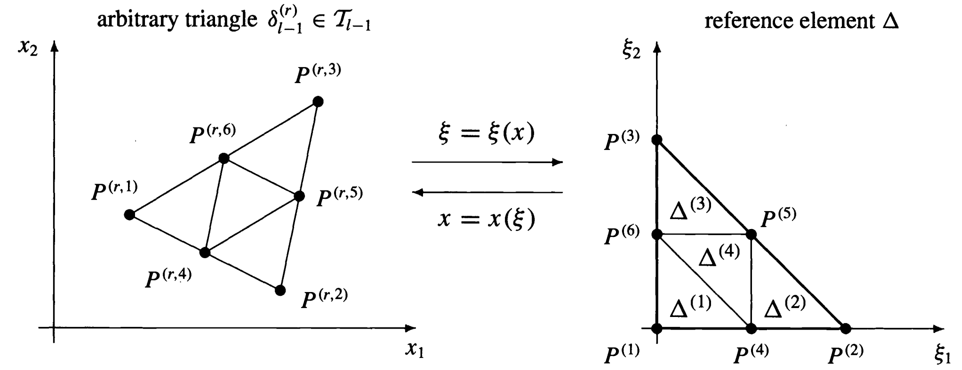
$$\frac{4}{3}K_l^{L,H} - \frac{1}{3}\tilde{K}_{l-1} = \begin{pmatrix} K_{l-1}^{L,N} & \frac{4}{3}K_{l,vm}^{L,H} \\ \frac{4}{3}K_{l,mv}^{L,H} & \frac{4}{3}K_{l,mm}^{L,H} \end{pmatrix} \tag{20}$$

and

$$K_l^{Q,H} = \begin{pmatrix} K_{l-1}^{L,N} & K_{l,vm}^{Q,H} \\ K_{l,mv}^{Q,H} & K_{l,mm}^{Q,H} \end{pmatrix}. \tag{21}$$

Taking into account that these matrices are symmetric, we have to prove that

$$\frac{4}{3}K_{l,vm}^{L,H} = K_{l,vm}^{Q,H} \quad \text{and} \quad \frac{4}{3}K_{l,mm}^{L,H} = K_{l,mm}^{Q,H}.$$



$$P^{(r,1)} = P^{(r,1)}(x_1^{(r,1)}, x_2^{(r,1)}),\qquad P^{(1)} = P^{(1)}(0,0),$$
$$P^{(r,2)} = P^{(r,2)}(x_1^{(r,2)}, x_2^{(r,2)}),\qquad P^{(2)} = P^{(2)}(1,0),$$
$$P^{(r,3)} = P^{(r,3)}(x_1^{(r,3)}, x_2^{(r,3)})\qquad P^{(3)} = P^{(3)}(0,1)$$

FIG. 1. *The mapping between the reference element Δ and an arbitrary element $\delta^{(r)}_{l-1}$.*

To do this we introduce some notation. The transformation $x = x(\xi)$,

$$\begin{aligned} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} &= \begin{pmatrix} x_1^{(r,2)} - x_1^{(r,1)} & x_1^{(r,3)} - x_1^{(r,1)} \\ x_2^{(r,2)} - x_2^{(r,1)} & x_2^{(r,3)} - x_2^{(r,1)} \end{pmatrix} \begin{pmatrix} \xi_1 \\ \xi_2 \end{pmatrix} + \begin{pmatrix} x_1^{(r,1)} \\ x_2^{(r,1)} \end{pmatrix} \\ &= J^{(r)}_{l-1} \begin{pmatrix} \xi_1 \\ \xi_2 \end{pmatrix} + \begin{pmatrix} x_1^{(r,1)} \\ x_2^{(r,1)} \end{pmatrix}, \end{aligned} \tag{22}$$

realizes the mapping of the reference element $\Delta = \{(\xi_1, \xi_2) : 0 \le \xi_1 \le 1,\ 0 \le \xi_2 \le 1,\ \xi_1 + \xi_2 \le 1\}$ onto an element $\delta^{(r)}_{l-1}$ of the triangulation $\mathcal{T}_{l-1}$ (see Fig. 1).

On the reference element Δ we define six shape functions φ_α, $\alpha = 1, 2, \ldots, 6$. In the case of the h-hierarchical basis we have

$$\begin{aligned} \tilde{\varphi}_1(\xi_1, \xi_2) &= \varphi_1(\xi_1, \xi_2) = 1 - \xi_1 - \xi_2, \\ \tilde{\varphi}_2(\xi_1, \xi_2) &= \varphi_2(\xi_1, \xi_2) = \xi_1, \\ \tilde{\varphi}_3(\xi_1, \xi_2) &= \varphi_3(\xi_1, \xi_2) = \xi_2, \\ \tilde{\varphi}_4(\xi_1, \xi_2) &= \varphi_4(\xi_1, \xi_2) = \begin{cases} 2\xi_1 & \text{in} \quad \Delta^{(1)}, \\ 2 - 2\xi_1 - 2\xi_2 & \text{in} \quad \Delta^{(2)}, \\ 0 & \text{in} \quad \Delta^{(3)}, \\ 1 - 2\xi_2 & \text{in} \quad \Delta^{(4)}, \end{cases} \\ \tilde{\varphi}_5(\xi_1, \xi_2) &= \varphi_5(\xi_1, \xi_2) = \begin{cases} 0 & \text{in} \quad \Delta^{(1)}, \\ 2\xi_2 & \text{in} \quad \Delta^{(2)}, \\ 2\xi_1 & \text{in} \quad \Delta^{(3)}, \\ 2\xi_1 + 2\xi_2 - 1 & \text{in} \quad \Delta^{(4)}, \end{cases} \\ \tilde{\varphi}_6(\xi_1, \xi_2) &= \varphi_6(\xi_1, \xi_2) = \begin{cases} 2\xi_2 & \text{in} \quad \Delta^{(1)}, \\ 0 & \text{in} \quad \Delta^{(2)}, \\ 2 - 2\xi_1 - 2\xi_2 & \text{in} \quad \Delta^{(3)}, \\ 1 - 2\xi_1 & \text{in} \quad \Delta^{(4)}, \end{cases} \end{aligned} \tag{23}$$

TABLE 1
The partial derivatives of the piecewise linear shape functions.

	$\frac{\partial\varphi_1}{\partial\xi_1}$	$\frac{\partial\varphi_1}{\partial\xi_2}$	$\frac{\partial\varphi_2}{\partial\xi_1}$	$\frac{\partial\varphi_2}{\partial\xi_2}$	$\frac{\partial\varphi_3}{\partial\xi_1}$	$\frac{\partial\varphi_3}{\partial\xi_2}$	$\frac{\partial\varphi_4}{\partial\xi_1}$	$\frac{\partial\varphi_4}{\partial\xi_2}$	$\frac{\partial\varphi_5}{\partial\xi_1}$	$\frac{\partial\varphi_5}{\partial\xi_2}$	$\frac{\partial\varphi_6}{\partial\xi_1}$	$\frac{\partial\varphi_6}{\partial\xi_2}$
$\Delta^{(1)}$	−1	−1	1	0	0	1	2	0	0	0	0	2
$\Delta^{(2)}$	−1	−1	1	0	0	1	−2	−2	0	2	0	0
$\Delta^{(3)}$	−1	−1	1	0	0	1	0	0	2	0	−2	−2
$\Delta^{(4)}$	−1	−1	1	0	0	1	0	−2	2	2	−2	0

TABLE 2
The partial derivatives of the quadratic shape functions.

	$\frac{\partial\psi_4}{\partial\xi_1}$	$\frac{\partial\psi_4}{\partial\xi_2}$	$\frac{\partial\psi_5}{\partial\xi_1}$	$\frac{\partial\psi_5}{\partial\xi_2}$	$\frac{\partial\psi_6}{\partial\xi_1}$	$\frac{\partial\psi_6}{\partial\xi_2}$
(0.5, 0)	0	−2	0	2	0	2
(0, 0.5)	2	0	2	0	−2	0
(0.5, 0.5)	−2	−2	2	2	−2	−2

and in the case of the p-hierarchical basis,

$$\begin{aligned}
&\tilde\varphi_1(\xi_1,\xi_2) = \varphi_1(\xi_1,\xi_2),\quad \tilde\varphi_2(\xi_1,\xi_2) = \varphi_2(\xi_1,\xi_2),\quad \tilde\varphi_3(\xi_1,\xi_2) = \varphi_3(\xi_1,\xi_2),\\
&\tilde\varphi_4(\xi_1,\xi_2) = \psi_4(\xi_1,\xi_2) = 4\xi_1(1-\xi_1-\xi_2),\\
&\tilde\varphi_5(\xi_1,\xi_2) = \psi_5(\xi_1,\xi_2) = 4\xi_1\xi_2,\\
&\tilde\varphi_6(\xi_1,\xi_2) = \psi_6(\xi_1,\xi_2) = 4\xi_2(1-\xi_1-\xi_2)\,.
\end{aligned} \tag{24}$$

In order to calculate the elements of the stiffness matrices we need the derivatives of the shape functions. For the h-hierarchical functions we get the partial derivatives given in Table 1.

For the computation of the matrix elements in the case of the p-hierarchical basis we use the following quadrature rule:

$$\int_\Delta \psi(\xi_1,\xi_2)\,d\xi \approx \sum_{k=1}^{3}\frac{1}{6}\,\psi(\xi^{(k)}) = \frac{1}{6}\left(\psi(0.5,0)+\psi(0,0.5)+\psi(0.5,0.5)\right), \tag{25}$$

which is exact for quadratic polynomials on Δ. Therefore we present in Table 2 the values of the partial derivatives of the functions ψ_4, ψ_5, and ψ_6 in the quadrature points $(0.5, 0)$, $(0, 0.5)$, and $(0.5, 0.5)$.

First we prove $\frac{4}{3}K^{L,H}_{l,mm} = K^{Q,H}_{l,mm}$. We have

$$\begin{aligned}
a(\tilde p^{(j)}_l,\tilde p^{(i)}_l) &= \int_\Omega \left(A\nabla_x\tilde p^{(j)}_l, \nabla_x\tilde p^{(i)}_l\right) dx\\
&= \sum_{r\in\omega^{(ij)}}\int_{\delta^{(r)}_{l-1}} \left(A\nabla_x\tilde p^{(j)}_l, \nabla_x\tilde p^{(i)}_l\right) dx,
\end{aligned} \tag{26}$$

where

$$\omega^{(ij)} = \left\{r\in\mathcal{J}_{l-1} : \tilde p^{(i)}_l \not\equiv 0 \text{ and } \tilde p^{(j)}_l \not\equiv 0 \text{ on } \delta^{(r)}_{l-1}\right\}.$$

Obviously, the index sets $\omega^{(ij)}$ are the same for both the h-hierarchical functions $\tilde{p}_l^{(i)} = p_l^{(i)}$ and the p-hierarchical functions $\tilde{p}_l^{(i)} = q_{l-1}^{(i)}$, $i = N_{l-1}+1, \dots, N_l$. Using the mapping to the reference element, it follows that

$$a(\tilde{p}_l^{(j)}, \tilde{p}_l^{(i)}) = \sum_{r\in\omega^{(ij)}} \int_\Delta \Big(A(J_{l-1}^{(r)})^{-T}\nabla_\xi \tilde{p}_l^{(j)}(x(\xi)), (J_{l-1}^{(r)})^{-T}\nabla_\xi \tilde{p}_l^{(i)}(x(\xi))\Big)\, |\det J_{l-1}^{(r)}|\, d\xi$$

$$= \sum_{r\in\omega^{(ij)}} \int_\Delta \Big(B\nabla_\xi \tilde{p}_l^{(j)}(x(\xi)), \nabla_\xi \tilde{p}_l^{(i)}(x(\xi))\Big)\, d\xi,$$

with $B = (J_{l-1}^{(r)})^{-1}A(J_{l-1}^{(r)})^{-T}|\det J_{l-1}^{(r)}|$. Note that the entries of A, $J_{l-1}^{(r)}$, and $|\det J_{l-1}^{(r)}|$ are constants.

For $j = i$ and $\tilde{p}_l^{(i)} = p_l^{(i)}$, that is, for the h-hierarchical basis, we have

$$(27)\qquad a(p_l^{(i)}, p_l^{(i)}) = \sum_{r\in\omega^{(ii)}} \sum_{k\in I(\alpha^{(r)})} \int_{\Delta^{(k)}} \left(B\nabla_\xi \varphi_{\alpha^{(r)}}, \nabla_\xi \varphi_{\alpha^{(r)}}\right) d\xi$$

$$= \sum_{r\in\omega^{(ii)}} \sum_{k\in I(\alpha^{(r)})} \frac{1}{8}\left(B\nabla_\xi \varphi_{\alpha^{(r)}}|_{\Delta^{(k)}}, \nabla_\xi \varphi_{\alpha^{(r)}}|_{\Delta^{(k)}}\right),$$

where $\alpha^{(r)}$ is the local number of the node $P^{(i)}(x_1^{(i)}, x_2^{(i)})$ in the triangle $\delta_{l-1}^{(r)}$, i.e.,

$$a^{(r)} = 4, 5, \text{ or } 6, \quad \text{and} \quad I(\alpha^{(r)}) = \left\{k \in \{1,2,3,4\} : \varphi_{\alpha^{(r)}} \not\equiv 0 \text{ on } \Delta^{(k)}\right\}.$$

Obviously, $I(4) = \{1,2,4\}$, $I(5) = \{2,3,4\}$, and $I(6) = \{1,3,4\}$. Therefore, in all the cases, we have exactly three terms.

If we use quadrature rule (25), we obtain for the case of the p-hierarchical basis

$$(28)\qquad a(q_{l-1}^{(i)}, q_{l-1}^{(i)}) = \sum_{r\in\omega^{(ii)}} \int_\Delta \left(B\nabla_\xi \psi_{\alpha^{(r)}}, \nabla_\xi \psi_{\alpha^{(r)}}\right) d\xi$$

$$= \sum_{r\in\omega^{(ii)}} \sum_{k=1}^{3} \frac{1}{6}\left(B\nabla_\xi \psi_{\alpha^{(r)}}(\xi^{(k)}), \nabla_\xi \psi_{\alpha^{(r)}}(\xi^{(k)})\right),$$

where $\xi^{(k)}$ are the quadrature nodes of formula (25).

Now we compare the summands in the sums over k in (27) and (28). If we examine the values of the partial derivatives $\partial\varphi_{\alpha^{(r)}}/\partial\xi_1$, $\partial\varphi_{\alpha^{(r)}}/\partial\xi_2$ in the triangles $\Delta^{(k)}$ and the values of the derivatives $\partial\psi_{\alpha^{(r)}}/\partial\xi_1$, $\partial\psi_{\alpha^{(r)}}/\partial\xi_2$ in the quadrature nodes (see Tables 1 and 2) then we can see that these summands differ only by the factor $\frac{4}{3}$. Therefore, we have

$$\frac{4}{3}a(p_l^{(i)}, p_l^{(i)}) = a(q_{l-1}^{(i)}, q_{l-1}^{(i)}) \qquad \text{for} \qquad i = N_{l-1}+1, \dots, N_l.$$

For $i \neq j$, $i, j = N_{l-1}+1, \dots, N_l$, we obtain

$$(29)\qquad a(q_{l-1}^{(j)}, q_{l-1}^{(i)}) = \sum_{r\in\omega^{(ij)}} \int_\Delta \left(B\nabla_\xi \varphi_{\beta^{(r)}}, \nabla_\xi \varphi_{\alpha^{(r)}}\right) d\xi,$$

where $\beta^{(r)}$ and $\alpha^{(r)}$ are the local numbers of the nodes $P^{(j)}$ and $P^{(i)}$ in the triangle $\delta_{l-1}^{(r)}$, respectively. Using again the quadrature formula (25) and the results from Table 2 we have

$$(30)\qquad a(q_{l-1}^{(j)}, q_{l-1}^{(i)}) = \begin{cases} \displaystyle\sum_{r\in\omega^{(ij)}} -\frac{1}{3}\left(B\begin{pmatrix}0\\2\end{pmatrix}, \begin{pmatrix}2\\2\end{pmatrix}\right) & \text{for} \quad \begin{matrix}\beta^{(r)}=4, \alpha^{(r)}=5,\\ \beta^{(r)}=5, \alpha^{(r)}=4,\end{matrix} \\ \displaystyle\sum_{r\in\omega^{(ij)}} \frac{1}{3}\left(B\begin{pmatrix}0\\2\end{pmatrix}, \begin{pmatrix}2\\0\end{pmatrix}\right) & \text{for} \quad \begin{matrix}\beta^{(r)}=4, \alpha^{(r)}=6,\\ \beta^{(r)}=6, \alpha^{(r)}=4,\end{matrix} \\ \displaystyle\sum_{r\in\omega^{(ij)}} -\frac{1}{3}\left(B\begin{pmatrix}2\\2\end{pmatrix}, \begin{pmatrix}2\\0\end{pmatrix}\right) & \text{for} \quad \begin{matrix}\beta^{(r)}=5, \alpha^{(r)}=6,\\ \beta^{(r)}=6, \alpha^{(r)}=5.\end{matrix} \end{cases}$$

For the h-hierarchical basis we get, with $I(\beta^{(r)}, \alpha^{(r)}) = I(\beta^{(r)}) \cap I(\alpha^{(r)})$,

$$\begin{aligned} a(p_l^{(j)}, p_l^{(i)}) &= \sum_{r\in\omega^{(ij)}} \sum_{k\in I(\beta^{(r)},\alpha^{(r)})} \int_{\Delta^{(k)}} \left(B\nabla_\xi \varphi_{\beta^{(r)}}, \nabla_\xi \varphi_{\alpha^{(r)}}\right) d\xi \\ (31)\qquad &= \sum_{r\in\omega^{(ij)}} \sum_{k\in I(\beta^{(r)},\alpha^{(r)})} \frac{1}{8}\left(B\nabla_\xi \varphi_{\beta^{(r)}}|_{\Delta^{(k)}}, \nabla_\xi \varphi_{\alpha^{(r)}}|_{\Delta^{(k)}}\right) \end{aligned}$$

$$= \begin{cases} \displaystyle\sum_{r\in\omega^{(ij)}} -\frac{1}{4}\left(B\begin{pmatrix}0\\2\end{pmatrix}, \begin{pmatrix}2\\2\end{pmatrix}\right) & \text{for} \quad \begin{matrix}\beta^{(r)}=4, \alpha^{(r)}=5,\\ \beta^{(r)}=5, \alpha^{(r)}=4,\end{matrix} \\ \displaystyle\sum_{r\in\omega^{(ij)}} \frac{1}{4}\left(B\begin{pmatrix}0\\2\end{pmatrix}, \begin{pmatrix}2\\0\end{pmatrix}\right) & \text{for} \quad \begin{matrix}\beta^{(r)}=4, \alpha^{(r)}=6,\\ \beta^{(r)}=6, \alpha^{(r)}=4,\end{matrix} \\ \displaystyle\sum_{r\in\omega^{(ij)}} -\frac{1}{4}\left(B\begin{pmatrix}2\\2\end{pmatrix}, \begin{pmatrix}2\\0\end{pmatrix}\right) & \text{for} \quad \begin{matrix}\beta^{(r)}=5, \alpha^{(r)}=6,\\ \beta^{(r)}=6, \alpha^{(r)}=5.\end{matrix} \end{cases}$$

Comparing (30) and (31) we see that $\frac{4}{3}a(p_l^{(j)}, p_l^{(i)}) = a(q_{l-1}^{(j)}, q_{l-1}^{(i)})$. Consequently, we have shown that

$$(32)\qquad \frac{4}{3}K_{l,mm}^{L,H} = K_{l,mm}^{Q,H}\,.$$

It remains to prove $\frac{4}{3}K_{l,vm}^{L,H} = K_{l,vm}^{Q,H}$. For $j = N_{l-1}+1, N_{l-1}+2, \ldots, N_l$, $i = 1, 2, \ldots, N_{l-1}$ we have

$$\begin{aligned} (33)\qquad a(p_l^{(j)}, p_{l-1}^{(i)}) &= \sum_{r\in\omega^{(ij)}} \int_\Delta (B\nabla_\xi \varphi_{\beta^{(r)}}, \nabla_\xi \varphi_{\alpha^{(r)}})\, d\xi \\ &= \sum_{r\in\omega^{(ij)}} \sum_{k\in I(\beta^{(r)})} \frac{1}{8}\left(B\nabla_\xi \varphi_{\beta^{(r)}}|_{\Delta^{(k)}}, \nabla_\xi \varphi_{\alpha^{(r)}}|_{\Delta^{(k)}}\right) \end{aligned}$$

and

$$\begin{aligned} (34)\qquad a(q_{l-1}^{(j)}, p_{l-1}^{(i)}) &= \sum_{r\in\omega^{(ij)}} \int_\Delta (B\nabla_\xi \psi_{\beta^{(r)}}, \nabla_\xi \varphi_{\alpha^{(r)}})\, d\xi \\ &= \sum_{r\in\omega^{(ij)}} \sum_{k=1}^{3} \frac{1}{6}\left(B\nabla_\xi \psi_{\beta^{(r)}}(\xi^{(k)}), \nabla_\xi \varphi_{\alpha^{(r)}}(\xi^{(k)})\right)\,. \end{aligned}$$

From Tables 1 and 2 we see again that the summands in the sums over k differ only by the factor $\frac{4}{3}$. Hence, $\frac{4}{3}a(p_l^{(j)}, p_{l-1}^{(i)}) = a(q_{l-1}^{(j)}, p_{l-1}^{(i)})$, i.e.,

$$(35)\qquad \frac{4}{3}K_{l,vm}^{L,H} = K_{l,vm}^{Q,H}\,,$$

and in an analogous way

$$\frac{4}{3}K_{l,mv}^{L,H} = K_{l,mv}^{Q,H}\,. \tag{36}$$

Combining the relations (20), (21), (32), (35), and (36) we obtain the statement of the lemma. □

In Lemma 2.2 we formulate the corresponding property for the right-hand side.

LEMMA 2.2. *Let*

$$\langle F, v\rangle = \int_\Omega f v\, dx + \int_{\Gamma_N} g_2 v\, ds\,,$$

where f is a piecewise constant function, i.e., constant over all triangles $\delta_{l-1}^{(r)} \in \mathcal{T}_{l-1}$, and g_2 a piecewise constant function, i.e., constant over $\partial\delta_{l-1}^{(r)} \cap \partial\Omega$. Then the following relation holds:

$$\underline{f}_l^{Q,H} = \frac{4}{3}\underline{f}_l^{L,H} - \frac{1}{3}\underline{\tilde{f}}_{l-1}\,, \qquad \underline{\tilde{f}}_{l-1} = \begin{pmatrix} \underline{f}_{l-1}^{L,N} \\ 0 \end{pmatrix}. \tag{37}$$

Proof. We have defined

$$\underline{f}_{l-1}^{L,N} = [f_{l-1}^{L,N,(i)}]_{i=1,2,\ldots,N_{l-1}}\,, \quad f_{l-1}^{L,N,(i)} = \langle F, p_{l-1}^{(i)}\rangle,$$

$$\underline{f}_l^{L,H} = [f_l^{L,H,(i)}]_{i=1,2,\ldots,N_l}\,, \quad f_l^{L,H,(i)} = \begin{cases} \langle F, p_{l-1}^{(i)}\rangle & \text{for} \quad i = 1,\ldots,N_{l-1}, \\ \langle F, p_l^{(i)}\rangle & \text{for} \quad i = N_{l-1}+1,\ldots,N_l\,, \end{cases}$$

$$\underline{f}_l^{Q,H} = [f_l^{Q,H,(i)}]_{i=1,2,\ldots,N_l}\,, \quad f_l^{Q,H,(i)} = \begin{cases} \langle F, p_{l-1}^{(i)}\rangle & \text{for} \quad i = 1,2,\ldots,N_{l-1}, \\ \langle F, q_{l-1}^{(i)}\rangle & \text{for} \quad i = N_{l-1}+1,\ldots,N_l\,. \end{cases}$$

Consequently,

$$\frac{4}{3}\underline{f}_l^{L,H} - \frac{1}{3}\underline{\tilde{f}}_{l-1} = \begin{pmatrix} \underline{f}_{l-1}^{L,N} \\ \frac{4}{3}\underline{f}_{l,m}^{L,H} \end{pmatrix} \quad \text{and} \quad \underline{f}_l^{Q,H} = \begin{pmatrix} \underline{f}_{l-1}^{L,N} \\ \underline{f}_{l,m}^{Q,H} \end{pmatrix}. \tag{38}$$

First we prove

$$\frac{4}{3}\int_\Omega f p_l^{(i)}\, dx = \int_\Omega f q_{l-1}^{(i)}\, dx \tag{39}$$

for $i = N_{l-1}+1,\ldots,N_l$. Using the notation from the proof of Lemma 2.1 we have

$$\begin{aligned} \int_\Omega f p_l^{(i)}\, dx = \sum_{r\in\omega^{(i)}} \int_{\delta_{l-1}^{(r)}} f p_l^{(i)}\, dx &= \sum_{r\in\omega^{(i)}} \int_\Delta f p_l^{(i)}(x(\xi))|\det J_{l-1}^{(r)}|\, d\xi \\ &= \sum_{r\in\omega^{(i)}} \sum_{k\in I(\alpha^{(r)})} \int_{\Delta^{(k)}} f\varphi_{\alpha^{(r)}}(\xi)|\det J_{l-1}^{(r)}|\, d\xi\,, \end{aligned}$$

where

$$\omega^{(i)} = \{r \in \mathcal{J}_{l-1} \;:\; p_l^{(i)} \not\equiv 0 \;\text{ on }\; \delta_{l-1}^{(r)}\},$$

$$I(\alpha^{(r)}) = \{k \in \{1,2,3,4\} \;:\; \varphi_{\alpha^{(r)}} \not\equiv 0 \;\text{ on }\; \Delta^{(k)}\},$$

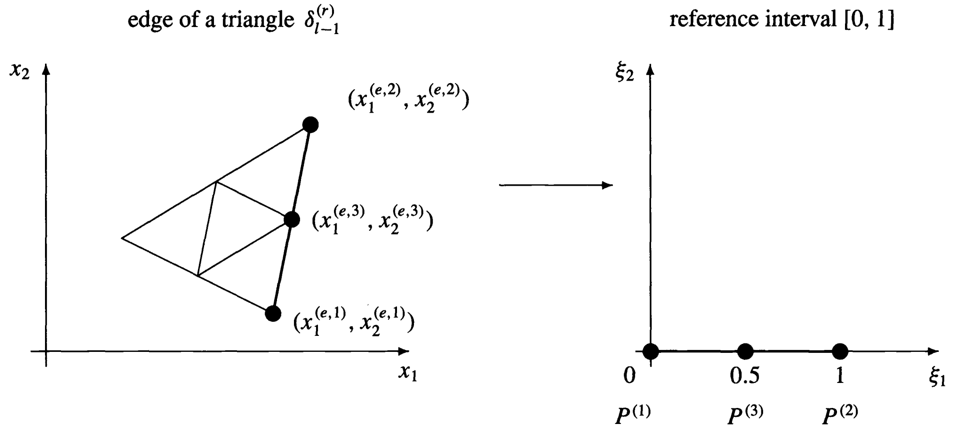

edge of a triangle $\delta_{l-1}^{(r)}$
reference interval [0, 1]
x_2
$(x_1^{(e,2)}, x_2^{(e,2)})$
$(x_1^{(e,3)}, x_2^{(e,3)})$
$(x_1^{(e,1)}, x_2^{(e,1)})$
x_1
ξ_2
0
0.5
1
ξ_1
$P^{(1)}$
$P^{(3)}$
$P^{(2)}$

FIG. 2. *The mapping between an edge of a triangle and the reference interval* [0, 1].

and $\alpha^{(r)}$ is the local number of the node $P^{(i)}$ in the triangle $\delta_{l-1}^{(r)}$. Computing the integrals over $\Delta^{(k)}$ we obtain

$$\int_\Omega f p_l^{(i)}\, dx = \sum_{r\in\omega^{(i)}} \sum_{k\in I(\alpha^{(r)})} \frac{1}{24} f\,|\det J_{l-1}^{(r)}| = \sum_{r\in\omega^{(i)}} \frac{1}{8} f\,|\det J_{l-1}^{(r)}|\,, \tag{40}$$

and using the quadrature formula (25) it follows that

$$\int_\Omega f q_{l-1}^{(i)}\, dx = \sum_{r\in\omega^{(i)}} \sum_{k=1}^{3} \frac{1}{6} f \psi_{\alpha^{(r)}}(\xi^{(k)})|\det J_{l-1}^{(r)}| = \sum_{r\in\omega^{(i)}} \frac{1}{6} f\,|\det J_{l-1}^{(r)}|\,, \tag{41}$$

i.e., the integrals in (40) and (41) differ by the factor $\frac{4}{3}$.

Next we show that

$$\frac{4}{3} \int_{\Gamma_N} g_2 p_l^{(i)}\, ds = \int_{\Gamma_N} g_2 q_{l-1}^{(i)}\, ds \tag{42}$$

for $i = N_{l-1} + 1, \ldots, N_l$. We have

$$\int_{\Gamma_N} g_2 p_l^{(i)}\, ds = \sum_{e\in E_{l-1}} \int_{\Gamma_{N,l-1}^{(e)}} g_2 p_l^{(i)}\, ds\,,$$

where $\Gamma_{N,l-1}^{(e)}$ is an edge of a triangle $\delta_{l-1}^{(r)}$, $r \in \mathcal{J}_{l-1}$, which is a part of the boundary Γ_N. We transform the last integral into an integral over the reference interval [0, 1].

This transformation is described by

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1^{(e,2)} - x_1^{(e,1)} \\ x_2^{(e,2)} - x_2^{(e,1)} \end{pmatrix} \xi_1 + \begin{pmatrix} x_1^{(e,1)} \\ x_2^{(e,1)} \end{pmatrix}$$

(see Fig. 2). On the reference interval the piecewise linear shape function $\varphi_3(\xi_1)$ is defined as follows:

$$\varphi_3(\xi_1) = \begin{cases} 2\xi_1 & \text{in} \quad [0, \frac{1}{2}), \\ 2 - 2\xi_1 & \text{in} \quad [\frac{1}{2}, 1], \end{cases}$$

and for the quadratic shape function $\psi_3(\xi_1)$ we have $\psi_3(\xi_1) = -4\xi_1^2 + 4\xi_1$.

With $\sigma = [(x_1^{(e,2)} - x_1^{(e,1)})^2 + (x_2^{(e,2)} - x_2^{(e,1)})^2]^{0.5}$ we obtain

$$\int_{\Gamma_N} g_2 p_l^{(i)}\, ds = \sum_{e \in E_{l-1}} \int_{\Gamma_{N,l-1}^{(e)}} g_2 p_l^{(i)}\, ds$$

$$(43) \qquad = \sum_{e \in E_{l-1}} \left\{ \int_0^{0.5} g_2 2\xi_1 \sigma\, d\xi_1 + \int_{0.5}^1 g_2 (2 - 2\xi_1)\sigma\, d\xi_1 \right\}$$

$$= \sum_{e \in E_{l-1}} g_2 \sigma \left\{ \frac{1}{4} + \frac{1}{4} \right\} = \sum_{e \in E_{l-1}} \frac{1}{2} g_2 \sigma$$

and

$$(44) \qquad \int_{\Gamma_N} g_2 q_{l-1}^{(i)}\, ds = \sum_{e \in E_{l-1}} \int_{\Gamma_{N,l-1}^{(e)}} g_2 q_{l-1}^{(i)}\, ds$$

$$= \sum_{e \in E_{l-1}} \int_0^1 g_2(-4\xi_1^2 + 4\xi_1)\sigma\, ds = \sum_{e \in E_{l-1}} \frac{2}{3} g_2 \sigma\,.$$

Again, both integrals differ only by the factor $\frac{4}{3}$. Combining the relations (38), (39), and (42) we get the statement of the lemma. □

THEOREM 2.3. *Under the assumptions of Lemmas* 2.1 *and* 2.2 *the FE systems of algebraic equations*

$$\left(\frac{4}{3}K_l^{L,H} - \frac{1}{3}\tilde{K}_{l-1}\right)\underline{u}_l^H = \left(\frac{4}{3}\underline{f}_l^{L,H} - \frac{1}{3}\tilde{\underline{f}}_{l-1}\right) \qquad \textit{and} \qquad K_l^{Q,H}\underline{u}_l^H = \underline{f}_l^{Q,H}$$

have the same solution.

Proof. The proof follows immediately from Lemmas 2.1 and 2.2. □

An analogous result can be proved for the FE systems in the nodal basis. Before we show this property, we state a lemma.

LEMMA 2.4. *Between the p-hierarchical and the quadratic nodal shape functions on the reference element it holds that*

$$(45) \qquad \Phi_\Delta^H = \Phi_\Delta^N S_\Delta\,,$$

where

$$S_\Delta = [S_\Delta^{(\alpha\beta)}]_{\alpha,\beta=1,\dots,6}\,, \quad S_\Delta^{(\alpha\beta)} = \begin{cases} 1 & \textit{for} \quad \alpha = \beta, \\ \frac{1}{2} & \textit{for} \quad (\alpha,\beta) = (4,1),(4,2), \\ & \qquad (\alpha,\beta) = (5,2),(5,3), \\ & \qquad (\alpha,\beta) = (6,3),(6,1), \\ 0 & \textit{otherwise}\,; \end{cases}$$

$$\Phi_\Delta^H = (\varphi_1(\xi_1,\xi_2),\ \varphi_2(\xi_1,\xi_2),\ \varphi_3(\xi_1,\xi_2),\ \psi_4(\xi_1,\xi_2),\ \psi_5(\xi_1,\xi_2),\ \psi_6(\xi_1,\xi_2)),$$

$$\Phi_\Delta^N = (\psi_1(\xi_1,\xi_2),\ \psi_2(\xi_1,\xi_2),\ \psi_3(\xi_1,\xi_2),\ \psi_4(\xi_1,\xi_2),\ \psi_5(\xi_1,\xi_2),\ \psi_6(\xi_1,\xi_2)),$$

with

$$(46) \qquad \begin{aligned} &\varphi_1(\xi_1,\xi_2) = 1 - \xi_1 - \xi_2\,, \quad \varphi_2(\xi_1,\xi_2) = \xi_1\,, \quad \varphi_3(\xi_1,\xi_2) = \xi_2\,, \\ &\psi_1(\xi_1,\xi_2) = 2\xi_1^2 + 2\xi_2^2 - 3\xi_1 - 3\xi_2 + 4\xi_1\xi_2 + 1\,, \\ &\psi_2(\xi_1,\xi_2) = 2\xi_1^2 - \xi_1\,, \quad \psi_3(\xi_1,\xi_2) = 2\xi_2^2 - \xi_2\,, \\ &\psi_4(\xi_1,\xi_2) = 4\xi_1(1 - \xi_1 - \xi_2)\,, \quad \psi_5(\xi_1,\xi_2) = 4\xi_1\xi_2\,, \\ &\psi_6(\xi_1,\xi_2) = 4\xi_2(1 - \xi_1 - \xi_2)\,. \end{aligned}$$

Proof. A simple calculation leads to

$$\begin{aligned}\varphi_1(\xi_1,\xi_2) &= \psi_1(\xi_1,\xi_2) + 0.5\ (\psi_4(\xi_1,\xi_2) + \psi_6(\xi_1,\xi_2)),\\ \varphi_2(\xi_1,\xi_2) &= \psi_2(\xi_1,\xi_2) + 0.5\ (\psi_4(\xi_1,\xi_2) + \psi_5(\xi_1,\xi_2)),\\ \varphi_3(\xi_1,\xi_2) &= \psi_3(\xi_1,\xi_2) + 0.5\ (\psi_5(\xi_1,\xi_2) + \psi_6(\xi_1,\xi_2)),\end{aligned}$$

and therefore (45) holds. □

LEMMA 2.5. *For the p-hierarchical and the quadratic nodal basis the relation*

$$\Phi_l^H = \Phi_l^N S_l \tag{47}$$

holds, where

$$\Phi_l^H = (p_{l-1}^{(1)}, p_{l-1}^{(2)}, \ldots, p_{l-1}^{(N_{l-1})}, q_{l-1}^{(N_{l-1}+1)}, \ldots, q_{l-1}^{(N_l)}), \tag{48}$$

$$\Phi_l^N = (q_{l-1}^{(1)}, q_{l-1}^{(2)}, \ldots, q_{l-1}^{(N_l)}), \tag{49}$$

$$S_l = [S_l^{(ij)}]_{i,j=1,2,\ldots,N_l}, \tag{50}$$

$$S_l^{(ij)} = \begin{cases} 1 & \textit{for } i = j,\ i,j = 1,2,\ldots,N_l,\\ \frac{1}{2} & \textit{for } j = i_1 \textit{ and } j = i_2,\ N_{l-1} < i \le N_l, \textit{ where } P^{(i)} \textit{ is the}\\ & \textit{midpoint of that edge which is given by the vertices}\\ & P^{(i_1)} \textit{ and } P^{(i_2)} \textit{ of a triangle of } \mathcal{T}_{l-1},\\ 0 & \textit{otherwise.}\end{cases} \tag{51}$$

Proof. The FE functions are defined element by element, i.e.,

$$\tilde{p}_l^{(i)}(x) = \begin{cases} \tilde{p}_\alpha^{(r)}(x) = \tilde{\varphi}_\alpha(\xi(x)), & x \in \delta_{l-1}^{(r)},\ r \in B_i,\\ 0 & \text{otherwise,}\end{cases}$$

where $\tilde{p}_l^{(i)}$ stands for one function from (48) or (49), $\tilde{\varphi}_\alpha$ stands for the corresponding shape function on the reference element Δ, i.e., for the corresponding function of (46), and $B_i = \{r \in \mathcal{J}_{l-1} : P^{(i)} \in \bar{\delta}_{l-1}^{(r)}\}$. Thus the statement of the lemma follows from Lemma 2.4 immediately. □

THEOREM 2.6. *Under the assumptions of Lemmas* 2.1 *and* 2.2 *the FE systems of algebraic equations*

$$\left(\frac{4}{3}K_l^{L,N} - \frac{1}{3}\tilde{K}_{l-1}\right)\underline{u}_l^N = \left(\frac{4}{3}\underline{f}_l^{L,N} - \frac{1}{3}\tilde{\underline{f}}_{l-1}\right) \quad \textit{and} \quad K_l^{Q,N}\underline{u}_l^N = \underline{f}_l^{Q,N}$$

have the same solution.

Proof. Using Lemma 2.5 we get for arbitrary vectors $\underline{u}_l,\ \underline{v}_l \in \mathbb{R}^{N_l}$

$$(K_l^{Q,H}\underline{u}_l, \underline{v}_l) = a(\Phi_l^H\underline{u}_l, \Phi_l^H\underline{v}_l) = a(\Phi_l^N S_l\,\underline{u}_l, \Phi_l^N S_l\,\underline{v}_l) = (S_l^T K_l^{Q,N} S_l\,\underline{u}_l, \underline{v}_l)$$

and

$$(\underline{f}_l^{Q,H}, \underline{v}_l) = \langle F, \Phi_l^H\underline{v}_l\rangle = \langle F, \Phi_l^N S_l\,\underline{v}_l\rangle = (S_l^T\underline{f}_l^{Q,N}, \underline{v}_l).$$

Therefore we have

$$K_l^{Q,H} = S_l^T K_l^{Q,N} S_l\,, \quad K_l^{Q,N} = S_l^{-T} K_l^{Q,H} S_l^{-1}, \tag{52}$$

$$\underline{f}_l^{Q,H} = S_l^T\underline{f}_l^{Q,N}, \quad \underline{f}_l^{Q,N} = S_l^{-T}\underline{f}_l^{Q,H}\,. \tag{53}$$

Furthermore, from Yserentant [24] we know that

$$K_l^{L,H} = S_l^T K_l^{L,N} S_l \quad \text{and} \quad \underline{f}_l^{L,H} = S_l^T \underline{f}_l^{L,N} \,. \tag{54}$$

From (52), (54), Lemma 2.1, and Lemma 2.2 it follows that

$$K_l^{Q,N} = S_l^{-T} K_l^{Q,H} S_l^{-1} \;=\; S_l^{-T} \left(\frac{4}{3} K_l^{L,H} - \frac{1}{3} \tilde{K}_{l-1} \right) S_l^{-1}$$

$$= \frac{4}{3} S_l^{-T} (S_l^T K_l^{L,N} S_l \,) S_l^{-1} - \frac{1}{3} S_l^{-T} \tilde{K}_{l-1}\, S_l^{-1} \;=\; \frac{4}{3} K_l^{L,N} - \frac{1}{3} \tilde{K}_{l-1} \tag{55}$$

and

$$\underline{f}_l^{Q,N} = S_l^{-T} \underline{f}_l^{Q,H} \;=\; S_l^{-T} \left(\frac{4}{3} \underline{f}_l^{L,H} - \frac{1}{3} \underline{\tilde{f}}_{l-1} \right)$$

$$= \frac{4}{3} S_l^{-T} S_l^T \underline{f}_l^{L,N} - \frac{1}{3} S_l^{-T} \underline{\tilde{f}}_{l-1} \;=\; \frac{4}{3} \underline{f}_l^{L,N} - \frac{1}{3} \underline{\tilde{f}}_{l-1} \,. \qquad \square \tag{56}$$

3. Multilevel algorithms with extrapolation. In this section we analyze a multigrid (MG) algorithm using FE discretizations with piecewise *linear* functions and an implicit extrapolation step. The iterates of this algorithm converge to the solution which we get by an FE discretization of problem (1) with piecewise *quadratic* functions. Additionally, we will use this algorithm as a preconditioner in the preconditioned conjugate gradient (PCCG) method.

First we introduce some notation.

- Smoothing procedure. Presmoothing $G_l^V(\underline{u}_l^{N,(j)}, K_l^{L,N}, \underline{f}_l^{L,N})$:

  Let the initial guess $\underline{u}_l^{N,(j)} = (\underline{u}_{l,v}^{N,(j)}, \underline{u}_{l,m}^{N,(j)})^T$ be given.

  Set $\underline{u}_{l,v}^{N,(j+1)} = \underline{u}_{l,v}^{N,(j)}$ and compute an approximate solution $\tilde{\underline{z}}_{l,m}$ of the system

$$K_{l,mm}^{L,N} \underline{z}_{l,m} = \underline{f}_{l,m}^{L,N} - K_{l,mv}^{L,N} \underline{u}_{l,v}^{N,(j+1)} - K_{l,mm}^{L,N} \underline{u}_{l,m}^{N,(j)} \tag{57}$$

  by means of an iterative method, starting with the zero-vector. We suppose that the error transmission operator of the method is of the type $M_{l,m} = (I_{l,m} - B_{l,mm}^{-1} K_{l,mm}^{L,N})$. Set $\underline{u}_l^{N,(j+1)} = (\underline{u}_{l,v}^{N,(j+1)}, \underline{u}_{l,m}^{N,(j)} + \tilde{\underline{z}}_{l,m})^T$.

  Postsmoothing $G_l^N(\underline{u}_l^{N,(j)}, K_l^{L,N}, \underline{f}_l^{L,N})$:

  We use the same algorithm; however, we suppose that the error transmission operator of the iterative method for solving the system (57) is of the type $M_{l,m} = (I_{l,m} - B_{l,mm}^{-T} K_{l,mm}^{L,N})$, so that the overall multigrid iteration operator becomes symmetric.
- Interpolation.

$$I_{l-1}^l \,:\, \mathbb{R}^{N_{l-1}} \to \mathbb{R}^{N_l} \,, \qquad I_{l-1}^l = \begin{pmatrix} I_{l,v} \\ S_{l,mv} \end{pmatrix} , \tag{58}$$

  where

$$(I_{l-1}^l)^{(ij)} = \begin{cases} 1 & \text{for } i = j,\ i,j = 1,2,\dots,N_{l-1}, \\ \frac{1}{2} & \text{for } j = i_1 \text{ and } j = i_2,\ N_{l-1} < i \le N_l, \text{ where } P^{(i)} \text{ is the} \\ & \text{midpoint of that edge which is given by the vertices} \\ & P^{(i_1)} \text{ and } P^{(i_2)} \text{ of a triangle from } \mathcal{T}_{l-1}, \\ 0 & \text{otherwise.} \end{cases} \tag{59}$$

- Restrictions.

$$I_l^{l-1} : \mathbb{R}^{N_l} \to \mathbb{R}^{N_{l-1}}, \quad I_l^{l-1} = (I_{l-1}^l)^T = (I_{l,v} \quad S_{l,mv}^T), \tag{60}$$

$$I_l^{l-1,inj} : \mathbb{R}^{N_l} \to \mathbb{R}^{N_{l-1}}, \quad I_l^{l-1,inj} = (I_{l,v} \quad 0). \tag{61}$$

Now we formulate the multigrid algorithm.

ALGORITHM 1. Let an initial guess $\underline{u}_l^{N,(k,0)}$ be given.

1. Presmoothing:

$$\underline{u}_l^{N,(k,1)} = G_l^V(\underline{u}_l^{N,(k,0)}, K_l^{L,N}, \underline{f}_l^{L,N}). \tag{62}$$

2. Coarse-grid correction.
   (a) Computation of the defect:

$$\underline{d}_{l-1}^{(k)} = \frac{4}{3} I_l^{l-1}(\underline{f}_l^{L,N} - K_l^{L,N}\underline{u}_l^{N,(k,1)}) - \frac{1}{3}(\underline{f}_{l-1}^{L,N} - K_{l-1}^{L,N} I_l^{l-1,inj}\underline{u}_l^{N,(k,1)}). \tag{63}$$

   (b) Solution of the system:

$$K_{l-1}^{L,N}\underline{w}_{l-1}^{(k)} = \underline{d}_{l-1}^{(k)} \tag{64}$$

   by means of μ iteration steps of a usual multigrid ($(l-1)$-grid) algorithm (see, e.g., [7]) which starts with the zero-vector and returns an approximate solution $\underline{\tilde{w}}_{l-1}^{(k)}$.

   (c) Computation of the correction:

$$\underline{u}_l^{N,(k,2)} = \underline{u}_l^{N,(k,1)} + I_{l-1}^l \underline{\tilde{w}}_{l-1}^{(k)}. \tag{65}$$

3. Postsmoothing:

$$\underline{u}_l^{N,(k,3)} = G_l^N(\underline{u}_l^{N,(k,2)}, K_l^{L,N}, \underline{f}_l^{L,N}). \tag{66}$$

Set $\underline{u}_l^{N,(k+1,0)} = \underline{u}_l^{N,(k,3)}$.

Before we present an alternative formulation of this algorithm, we analyze the smoothing step and the computation of the defect.

- The essential operation in the smoothing step is the approximate solution of system (57). Obviously, we can replace equation (57) by

$$\frac{4}{3}K_{l,mm}^{L,N}\underline{z}_{l,m} = \frac{4}{3}\underline{f}_{l,m}^{L,N} - \frac{4}{3}K_{l,mv}^{L,N}\underline{u}_{l,v}^{N,(j+1)} - \frac{4}{3}K_{l,mm}^{L,N}\underline{u}_{l,m}^{N,(j)}\ . \tag{67}$$

  Using the relations (55) and (56) in the proof of Theorem 2.6 we get the equivalence of relation (67) to

$$K_{l,mm}^{Q,N}\underline{z}_{l,m} = \underline{f}_{l,m}^{Q,N} - K_{l,mv}^{Q,N}\underline{u}_{l,v}^{N,(j+1)} - K_{l,mm}^{Q,N}\underline{u}_{l,m}^{N,(j)}\ . \tag{68}$$

- Step 2(a) in Algorithm 1 can be formulated in terms of the quadratic nodal basis. We have

$$\frac{4}{3} I_l^{l-1}(\underline{f}_l^{L,N} - K_l^{L,N} \underline{u}_l^{N,(k,1)}) - \frac{1}{3}(\underline{f}_{l-1}^{L,N} - K_{l-1}^{L,N} I_l^{l-1,inj} \underline{u}_l^{N,(k,1)})$$

$$= I_l^{l-1}\left[\left(\frac{4}{3}\underline{f}_l^{L,N} - \frac{1}{3}\underline{\tilde{f}}_{l-1}\right) - \left(\frac{4}{3}K_l^{L,N} - \tilde{K}_{l-1}\right)\underline{u}_l^{N,(k,1)}\right] \tag{69}$$

$$= I_l^{l-1}(\underline{f}_l^{Q,N} - K_l^{Q,N} \underline{u}_l^{N,(k,1)}).$$

Because of the equivalence of the relations (57) and (68) we can replace in Algorithm 1 the smoothing steps (62) and (66) by the equivalent steps

$$\underline{u}_l^{N,(k,1)} = G_l^V(\underline{u}_l^{N,(k,0)}, K_l^{Q,N}, \underline{f}_l^{Q,N}) \quad \text{and} \quad \underline{u}_l^{N,(k,3)} = G_l^N(\underline{u}_l^{N,(k,2)}, K_l^{Q,N}, \underline{f}_l^{Q,N}).$$

Furthermore, we can see from equation (69) that the computation of the defect (63) is equivalent to

$$\underline{d}_{l-1}^{(k)} = I_l^{l-1}(\underline{f}_l^{Q,N} - K_l^{Q,N} \underline{u}_l^{N,(k,1)}).$$

Therefore, Algorithm 1 can be interpreted as a multigrid algorithm for solving the system $K_l^{Q,N} \underline{u}_l = \underline{f}_l^{Q,N}$ of algebraic finite element equations resulting from a discretization with piecewise quadratic functions.

We can also formulate Algorithm 1 in terms of the h- or p-hierarchical basis. Because of

$$\underline{u}_l^{N,(j)} = S_l \underline{u}_l^{H,(j)}, \text{ i.e., } \underline{u}_{l,v}^{N,(j)} = \underline{u}_{l,v}^{H,(j)}, \quad \underline{u}_{l,m}^{N,(j)} = S_{l,mv} \underline{u}_{l,v}^{H,(j)} + \underline{u}_{l,m}^{H,(j)},$$

with S_l defined in (50), $\underline{f}_{l,m}^{L,H} = \underline{f}_{l,m}^{L,N}$, and $K_{l,mm}^{L,H} = K_{l,mm}^{L,N}$, it follows from (67) that

$$\frac{4}{3} K_{l,mm}^{L,H} \underline{z}_{l,m} = \frac{4}{3} \underline{f}_{l,m}^{L,H} - \frac{4}{3} K_{l,mv}^{L,N} \underline{u}_{l,v}^{H,(j+1)} - \frac{4}{3} K_{l,mm}^{L,N}(S_{l,mv} \underline{u}_{l,v}^{H,(j)} + \underline{u}_{l,m}^{H,(j)}).$$

Taking into account that we set $\underline{u}_{l,v}^{H,(j+1)} = \underline{u}_{l,v}^{H,(j)}$ in the smoothing procedure defined at the beginning of this section and that $K_{l,mv}^{L,H} = K_{l,mv}^{L,N} + K_{l,mm}^{L,N} S_{l,mv}$ is valid (see also relation (54)) we get

$$\frac{4}{3} K_{l,mm}^{L,H} \underline{z}_{l,m} = \frac{4}{3} \underline{f}_{l,m}^{L,H} - \frac{4}{3} K_{l,mv}^{L,H} \underline{u}_{l,v}^{H,(j+1)} - \frac{4}{3} K_{l,mm}^{L,H} \underline{u}_{l,m}^{H,(j)} \tag{70}$$

or the equivalent relation,

$$K_{l,mm}^{Q,H} = \underline{f}_{l,m}^{Q,H} - K_{l,mv}^{Q,H} \underline{u}_{l,v}^{H,(j+1)} - K_{l,mm}^{Q,H} \underline{u}_{l,m}^{H,(j)}. \tag{71}$$

Using the relation (54) a simple calculation leads to the equivalence of the coarse-grid correction step in Algorithm 1,

$$\underline{u}_l^{N,(k,2)} = \underline{u}_l^{N,(k,1)} + I_{l-1}^l(I_{l-1} - M_{l-1}^{\mu})(K_{l-1}^{L,N})^{-1} I_l^{l-1}\left[\left(\frac{4}{3}\underline{f}_l^{L,N} - \frac{1}{3}\underline{\tilde{f}}_{l-1}\right)\right.$$
$$\left. - \left(\frac{4}{3}K_l^{L,N} - \frac{1}{3}\tilde{K}_{l-1}\right)\underline{u}_l^{N,(k,1)}\right],$$

to the coarse-grid correction step

$$\underline{u}_l^{H,(k,2)} = \underline{u}_l^{H,(k,1)}$$
$$+ \tilde{I}_{l-1}^l(I_{l-1} - M_{l-1}^{\mu})(K_{l-1}^{L,N})^{-1}\left[\underline{f}_{l-1}^{L,N} - \left(K_{l-1}^{L,N} \underline{u}_{l,v}^{H,(k,1)} - \frac{4}{3} K_{l,vm}^{L,H} \underline{u}_{l,m}^{H,(k,1)}\right)\right] \tag{72}$$

in the h-hierarchical basis, where $\tilde{I}^{l}_{l-1}\underline{v}_{l-1} = (\underline{v}_{l-1} \quad 0)^T$ for all $\underline{v}_{l-1} \in \mathbb{R}^{N_{l-1}}$, I_{l-1} is the identity operator in the space $\mathbb{R}^{N_{l-1}}$, and M_{l-1} denotes the error transmission operator of the $(l-1)$-grid method for solving the coarse-grid system (64). In the p-hierarchical basis this becomes

$$\begin{aligned}\underline{u}_l^{H,(k,2)} &= \underline{u}_l^{H,(k,1)} \\ &+ \tilde{I}^{l}_{l-1}(I_{l-1} - M^{\mu}_{l-1})(K^{L,N}_{l-1})^{-1}\left[\underline{f}^{L,N}_{l-1} - \left(K^{L,N}_{l-1}\underline{u}^{H,(k,1)}_{l,v} - K^{Q,H}_{l,vm}\underline{u}^{H,(k,1)}_{l,m}\right)\right].\end{aligned} \tag{73}$$

Instead of using the initial guess $\underline{u}_l^{N,(k,0)}$, we set $\underline{u}_l^{H,(k,0)} = S_l^{-1}\underline{u}_l^{N,(k,0)}$ and replace the smoothing steps by the equivalent steps

$$G_l^V(\underline{u}_l^{H,(j)}, K_l^{L,H}, \underline{f}_l^{L,H}), \quad G_l^N(\underline{u}_l^{H,(j)}, K_l^{L,H}, \underline{f}_l^{L,H}) \qquad \text{or}$$

$$G_l^V(\underline{u}_l^{H,(j)}, K_l^{Q,H}, \underline{f}_l^{Q,H}), \quad G_l^N(\underline{u}_l^{H,(j)}, K_l^{Q,H}, \underline{f}_l^{Q,H})$$

as well as the coarse-grid correction step by the step (72) or (73), respectively. Thus we obtain the formulation of Algorithm 1 in the h- and p-hierarchical basis.

According to these interpretations we can formulate Algorithm 1 in a more abstract form. If we use a decomposition of the FE space V_l, i.e.,

$$V_l = V_l^{Q,N} = V_l^{Q,H} = V_{l-1} + T_l \ , \qquad T_l = \operatorname{span}\{q^{(i)}_{l-1}\ , \ i = N_{l-1}+1, \ldots, N_l\} \tag{74}$$

we get the following equivalent algorithm.

ALGORITHM 1′. Let an initial guess $u_l^{(k,0)} \in V_l$ be given.

1. Presmoothing.

$$\begin{aligned}&\text{Determine} \quad u_l^{(k,1)} \in u_l^{(k,0)} + T_l \ : \|u_l^{(k,1)} - u_{l,*}^{(k,1)}\| \le \rho_1 \|u_l^{(k,0)} - u_{l,*}^{(k,1)}\| \\ &\text{where} \quad u_{l,*}^{(k,1)} \in u_l^{(k,0)} + T_l \ : a(u_{l,*}^{(k,1)}, v) = \langle F, v\rangle \quad \text{for all } v \in T_l\,.\end{aligned} \tag{75}$$

2. Coarse-grid correction.

$$\begin{aligned}&\text{Determine} \quad u_l^{(k,2)} \in u_l^{(k,1)} + V_{l-1} : \|u_l^{(k,2)} - u_{l,*}^{(k,2)}\| \le \rho_2 \|u_l^{(k,1)} - u_{l,*}^{(k,2)}\| \\ &\text{where} \quad u_{l,*}^{(k,2)} \in u_l^{(k,1)} + V_{l-1} : a(u_{l,*}^{(k,2)}, v) = \langle F, v\rangle \quad \text{for all } v \in V_{l-1}.\end{aligned} \tag{76}$$

3. Postsmoothing.

$$\begin{aligned}&\text{Determine} \quad u_l^{(k,3)} \in u_l^{(k,2)} + T_l \ : \|u_l^{(k,3)} - u_{l,*}^{(k,3)}\| \le \rho_3 \|u_l^{(k,2)} - u_{l,*}^{(k,3)}\| \\ &\text{where} \quad u_{l,*}^{(k,3)} \in u_l^{(k,2)} + T_l \ : a(u_{l,*}^{(k,3)}, v) = \langle F, v\rangle \quad \text{for all } v \in T_l\,.\end{aligned} \tag{77}$$

Set $u_l^{(k+1,0)} = u_l^{(k,3)}$.

Depending on the choice of the basis in the space V_l we get from Algorithm 1′ the multigrid algorithm for solving the FE system of algebraic equations in terms of the piecewise quadratic nodal basis or the p-hierarchical basis.

In [21], Schieweck has proved the following convergence result for this type of multigrid algorithm:

$$\|u_l^{(k+1,0)} - u_l\| \le \eta\, \|u_l^{(k,0)} - u_l\|\,, \tag{78}$$

where

$$\eta = \rho_2 + (1-\rho_2)[\rho_1 + (1-\rho_1)\gamma][\rho_3 + (1-\rho_3)\gamma]\,, \tag{79}$$

$\|.\|^2 = a(.,.)$, and u_l is the solution of the problem

$$\text{find } u_l \in V_l \ : \ a(u_l, v_l) = \langle F, v_l\rangle \quad \text{for all} \quad v_l \in V_l\,,$$

and γ is the constant in the strengthened Cauchy inequality

$$|a(v_l, w_{l-1})| \le \gamma \|v_l\| \, \|w_{l-1}\| \quad \text{for all} \quad v_l \in T_l, \ \text{for all} \quad w_{l-1} \in V_{l-1}\,. \tag{80}$$

Using this result we can prove the following convergence theorem for Algorithm 1.

THEOREM 3.1. *Let the smoothing procedures, the restriction, and the interpolation operators be defined as at the beginning of this section and let the assumptions of Lemmas* 2.1 *and* 2.2 *be fulfilled. Then the following occur.*

(i) *The iterates of Algorithm* 1 *converge to the solution which we get by an FE discretization of problem* (1) *with piecewise quadratic functions.*

(ii) *The convergence estimate*

$$\|\underline{u}_l^{(k+1,0)} - \underline{u}_l\|_* \le \eta \|\underline{u}_l^{(k,0)} - \underline{u}_l\|_* \tag{81}$$

holds, where $\|.\|_*^2 = ((\frac{4}{3}K_l^{L,N} - \frac{1}{3}\tilde{K}_{l-1}).,.)$ *and* $\underline{u}_l$ *is the solution of the system of algebraic FE equations*

$$\left(\frac{4}{3}K_l^{L,N} - \frac{1}{3}\tilde{K}_{l-1}\right)\underline{u}_l = \left(\frac{4}{3}\underline{f}_l^{L,N} - \frac{1}{3}\underline{\tilde{f}}_{l-1}\right).$$

The convergence rate η *depends on the number of iteration steps for solving the systems* (57), *on the convergence rate of the* $(l-1)$*-grid algorithm used in step* 2(b), *and on the constant in the strengthened Cauchy inequality* (80).

Proof.

(i) This follows from the interpretation of Algorithm 1 as a multigrid algorithm for solving the FE system $K_l^{Q,N}\underline{u}_l = \underline{f}_l^{Q,N}$, immediately.

(ii) The convergence estimate (81) follows from estimate (79), because Algorithm 1 is equivalent to Algorithm 1′.

From [1] we know that the matrices $K_{l,mm}^{L,N}$ and $K_{l,mm}^{Q,N}$ have a condition number which is independent of the discretization parameter. Therefore ρ_1 and ρ_3 in (75) and (77), respectively, do not depend on the discretization parameter. If additionally the convergence rate of the $(l-1)$-grid algorithm for solving the system (64) is independent of the discretization parameter h_{l-1}, then we get an h_l-independent convergence rate η of Algorithm 1. □

Remark 3.1. The strengthened Cauchy inequality (80) for various bilinear forms $a(.,.)$ was analyzed by many authors [1], [2], [5], [9], [12], [14], [21], [23]. Maitre and Musy [14] calculated the constant γ for bilinear forms corresponding to scalar partial differential equations of second order. Jung [9] and Jung, Langer, and Semmler [12] studied the dependence of γ on the Poisson ratio for linear elasticity problems in two and three dimensions.

Remark 3.2. For different bilinear forms, the dependence of ρ_1 and ρ_3 on problem-specific parameters is studied in [9], [12], and [21].

Remark 3.3. The statements of Theorem 3.1 can also be proved for Algorithm 1 applied to FE equations resulting from the discretization of plane linear elasticity problems. To get these results we must prove the statements of Lemmas 2.1 and 2.2 for the related matrices $K_l^{L,H}$, $K_{l-1}^{L,N}$, and $K_l^{Q,H}$. These proofs are similar to the proofs given in §2. In §4 we will show some numerical experiments for plane linear elasticity problems.

Remark 3.4. We can also use Algorithm 1 as a preconditioner. The starting point is the PCCG method for solving the system of algebraic equations

$$\left(\frac{4}{3}K_l^{L,N} - \frac{1}{3}\tilde{K}_{l-1}\right)\underline{u}_l = \left(\frac{4}{3}\underline{f}_l^{L,N} - \frac{1}{3}\underline{\tilde{f}}_{l-1}\right) . \tag{82}$$

Since the matrix of the system of equations (82) is only used for matrix by vector multiplications within the PCCG method it is not necessary to assemble the matrix $\left(\frac{4}{3}K_l^{L,N} - \frac{1}{3}\tilde{K}_{l-1}\right)$. Also, the right-hand side is needed for the computation of the defect in the initial step of the PCCG method only. Therefore, we can perform all operations of the PCCG method using the matrices $K_l^{L,N}$, $K_{l-1}^{L,N}$ and the right–hand sides $\underline{f}_l^{L,N}$ and $\underline{f}_{l-1}^{L,N}$. A priori we choose the matrix $\tilde{C}_l = \left(\frac{4}{3}K_l^{L,N} - \frac{1}{3}\tilde{K}_{l-1}\right)$ as preconditioner and solve the preconditioning systems $\tilde{C}_l\underline{w}_l = \underline{r}_l$ within the PCCG algorithm by means of Algorithm 1. This approach we can interpret as a preconditioning with the matrix

$$C_l = \left(\frac{4}{3}K_l^{L,N} - \frac{1}{3}\tilde{K}_{l-1}\right)\left(I_l - M_l^m\right)^{-1} , \tag{83}$$

where M_l^m is the error transmission operator of Algorithm 1. We have to check whether the matrix C_l is symmetric and positive definite. In [11] some conditions for the smoothing procedures, the restriction, and the interpolation operators are given, which guarantee these properties. It is easy to verify these conditions for our case.

4. Numerical results. In this section we want to demonstrate that the iterates of Algorithm 1 converge to the FE solution which we would obtain by a discretization of problem (1) with piecewise quadratic functions. Furthermore, we show that the convergence rate of Algorithm 1 is independent of the discretization parameter. We compare Algorithm 1 with a multigrid algorithm applied to FE equations resulting from a discretization with quadratic elements.

The primary purpose of these experiments is to demonstrate the validity of our previous analysis and to show that it leads to efficient algorithms. This, however, means that the algorithms are not necessarily optimal from a more pragmatic point of view. For example, our stopping criterion (see (85) below) forces the algorithm to compute solutions much better than truncation error accuracy. Our goal is not to compute an acceptable solution at minimal cost, but to show the numerical equivalence of the algorithmic variants discussed above.

For the same reason we have introduced the smoothing procedure (62) which is restricted to the fine-grid nodes excluding nodes belonging to the coarser grid. With this smoother the different algorithms converge to exactly the same result. In practice, a τ-extrapolation algorithm may be more efficient when used with regular multigrid smoothing. This, however, would introduce perturbations that change the solution and is therefore not studied here, because a careful analysis similar to Hackbusch [7] or Rüde [17] is beyond the scope of the present paper. For an analysis of various multigrid τ-extrapolation algorithms and interesting numerical experiments with different smoothing procedures we also refer the reader to Bernert [3].

Finally, we remark that our full multigrid strategy could be improved by using τ-extrapolation to compute the coarser level solutions and then use higher order interpolation to find a

TABLE 3
Comparison of the CPU time needed for the generation of the FE systems.

Level l	Number of degrees of freedom	Number of triangles of $\mathcal{T}_{l-1}$	CPU time for the generation of	
			$K_{l-1}^{L,N}$, $K_l^{L,N}$, $\underline{f}_{l-1}^{L,N}$, $\underline{f}_l^{L,N}$	$K_l^{Q,H}$, $\underline{f}_l^{Q,H}$
3	49	32	0.007 sec	0.011 sec
4	225	128	0.029 sec	0.044 sec
5	961	512	0.118 sec	0.178 sec
6	3969	2048	0.473 sec	0.713 sec
7	16129	8192	1.892 sec	2.851 sec

more accurate initial guess for each new level. This has not been implemented for reasons of simplicity.

All algorithms have been implemented within the multigrid package FEMGP [10], [22]. The computations were performed on a PC 80486 (33 MHz) using the LAHEY-Fortran compiler.

Let us first consider the problem: find $u \in H_0^1(\Omega)$ such that

$$\int_\Omega (A\nabla_x u, \nabla_x v)\,dx = \int_\Omega f v\,dx \tag{84}$$

holds for all $v \in H_0^1(\Omega)$, where $\Omega = (0,1)\times(0,1)$, $A = \left(\begin{smallmatrix}4 & 4\\ 4 & 5\end{smallmatrix}\right)$, and $f = \pi^2(9\sin\pi x\sin\pi y - 8\cos\pi x\cos\pi y)$. The exact solution of this problem is $u = \sin\pi x\sin\pi y$.

In §3 we have seen that we can give equivalent formulations of Algorithm 1 in terms of the quadratic nodal basis, the h-hierarchical basis, and the p-hierarchical basis, respectively. A detailed analysis of the arithmetical work needed for the generation of the stiffness matrices and right-hand sides as well as for a matrix by vector multiplication shows that the implementation of Algorithm 1 in terms of the linear nodal basis or the equivalent algorithm in the p-hierarchical basis will lead to the best algorithms with respect to the necessary CPU-time (see also [13]). For example, for the generation of the matrices and right-hand sides, we need

$K_{l-1}^{L,N}$, $K_l^{L,N}$, $\underline{f}_{l-1}^{L,N}$, $\underline{f}_l^{L,N}$:	67 R_{l-1} additions and 51 R_{l-1} multiplications,
$K_l^{L,H}$, $\underline{f}_l^{L,H}$:	105 R_{l-1} additions and 169 R_{l-1} multiplications,
$K_l^{Q,N}$, $\underline{f}_l^{Q,N}$, $K_{l-1}^{L,N}$:	111 R_{l-1} additions and 146 R_{l-1} multiplications,
$K_l^{Q,H}$, $\underline{f}_l^{Q,H}$:	81 R_{l-1} additions and 120 R_{l-1} multiplications,

where R_{l-1} denotes the number of triangles in the triangulation $\mathcal{T}_{l-1}$. Table 3 demonstrates this for the linear nodal basis and the p-hierarchical basis.

Now we compare Algorithm 1 with the equivalent Algorithm 1′ in terms of the p-hierarchical basis, in the following, which is called Algorithm 1′(H, Q). Within Algorithm 1 and Algorithm 1′(H, Q) we used for the presmoothing (62) two iteration steps of the lexicographically forward Gauss–Seidel method, one iteration step of an $(l-1)$-grid algorithm for solving the coarse-grid system (64), and two iteration steps of the lexicographically backward Gauss–Seidel method for the postsmoothing (66). The initial guess was obtained by a full multigrid strategy using, on the levels $k = 1, 2, \dots, l-1$, a usual multigrid algorithm for solving the corresponding FE equations in the linear nodal basis, i.e., we performed within these k-grid algorithms two W-cycles with two Gauss–Seidel steps lexicographically forward in the presmoothing step and two Gauss–Seidel steps lexicographically backward in the postsmoothing step. For the k-grid algorithms, $k = 1, 2, \dots, l-1$, we observed a convergence rate of

TABLE 4
Comparison of Algorithm 1, *Algorithm* $1'(H, Q)$, *and the MG*(1)*-PCCG method.*

l	Algorithm 1		Algorithm $1'(H, Q)$		MG(1)-PCCG method	
	Number of iterations	CPU time	Number of iterations	CPU time	Number of iterations	CPU time
3	13	0.11 sec	13	0.11 sec	5	0.06 sec
4	14	0.54 sec	14	0.55 sec	6	0.28 sec
5	14	2.36 sec	14	2.41 sec	6	1.15 sec
6	14	9.83 sec	14	10.48 sec	6	4.83 sec
7	14	41.58 sec	14	44.33 sec	6	19.83 sec

0.085. The iteration of Algorithm 1 or Algorithm $1'(H, Q)$ is terminated when the defect criterion

$$\frac{\left\| \left(\frac{4}{3}\underline{f}_l^{L,N} - \frac{1}{3}\underline{\tilde{f}}_{l-1}\right) - \left(\frac{4}{3}K_l^{L,N} - \frac{1}{3}\tilde{K}_{l-1}\right)\underline{u}_l^{N,(k+1,0)} \right\|}{\left\| \left(\frac{4}{3}\underline{f}_l^{L,N} - \frac{1}{3}\underline{\tilde{f}}_{l-1}\right) - \left(\frac{4}{3}K_l^{L,N} - \frac{1}{3}\tilde{K}_{l-1}\right)\underline{u}_l^{N,(0,0)} \right\|} \leq 10^{-4} \tag{85}$$

or

$$\|\underline{f}_l^{Q,N} - K_l^{Q,N}\underline{u}_l^{N,(k+1,0)}\| \leq 10^{-4} \, \|\underline{f}_l^{Q,N} - K_l^{Q,N}\underline{u}_l^{N,(0,0)}\| \tag{86}$$

is fulfilled, where $\| \, . \, \|$ denotes the Euclidian norm in the space $\mathbb{R}^{N_l}$, and where $\underline{u}_l^{N,(0,0)}$ is the initial guess. In Table 4 the number of iterations and the CPU time needed by the application of Algorithm 1 and Algorithm $1'(H, Q)$ are given. The results show that the number of iterations is independent of the discretization parameter. In this example the constant in the strengthened Cauchy inequality (80) is 0.98. With the observed convergence rate of 0.085 of the $(l-1)$-grid method and this constant in the strengthened Cauchy inequality we conclude that the bound on the convergence rate of Algorithm 1 cannot be better than 0.96. Note that for the Laplace operator the constant would be $\sqrt{2/3}$ (see Maitre and Musy [14]) and would therefore lead to much less pessimistic and obviously also more realistic convergence estimates.

If we use one iteration step of Algorithm 1 as preconditioner in the PCCG method for solving the system

$$\left(K_l^{L,N} - \frac{1}{3}\tilde{K}_{l-1}\right)\underline{u}_l = \left(\underline{f}_l^{L,N} - \frac{1}{3}\underline{\tilde{f}}_{l-1}\right), \tag{87}$$

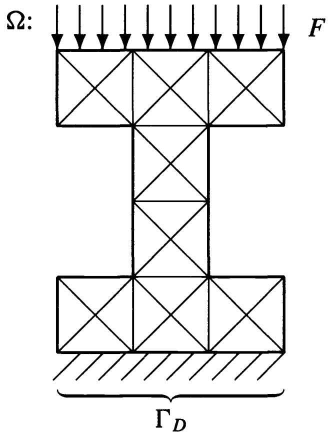
we get an algorithm with better convergence, the so-called MG(1)-PCCG method (see also Remark 3.4). The iteration of the MG(1)-PCCG method is again terminated when the criterion (85) is fulfilled.

Finally, we compare the discretization errors $\|u - u_l^L\|_1$ and $\|u - u_l^Q\|_1$ in the H^1-norm (see Table 5). Here u_l^L denotes the FE solution obtained by means of Algorithm 1, and u_l^Q the FE solution by a discretization with piecewise quadratic functions. We remark that in our example the right-hand side f is not constant on triangles $\delta_{l-1}^{(r)}$, which we had assumed in the proof of Theorems 2.3 and 2.6. Therefore, in our example the right-hand sides $\left(\frac{4}{3}\underline{f}_l^{L,N} - \frac{1}{3}\underline{\tilde{f}}_{l-1}\right)$ and $\underline{f}_l^{Q,N}$ are not identical. But the discretization errors are almost the same.

As a second example we consider a linear elasticity problem: find the displacement field $u = (u_1, u_2)^T \in V_0$, such that

TABLE 5
Comparison of the discretization errors.

Level l	$\|u-u_l^L\|_1$	$\|u-u_l^Q\|_1$
3	0.1306	0.1426
4	0.3347-01	0.3481-01
5	0.8426-02	0.8539-02
6	0.2110-02	0.2118-02
7	0.5278-03	0.5283-03



$$
\begin{aligned}
E &= 196\ GPa\\
\nu &= 0.3\\
g_{2,1} &= 0\\
g_{2,2} &= \begin{cases} F = 1000\ N & \text{on the upper part of the boundary}\\ 0 & \text{otherwise}\end{cases}
\end{aligned}
$$

FIG. 3. *Shape of the domain and data for the test problem.*

$$
(88)\qquad
\begin{aligned}
&\frac{E}{1+\nu}\int_\Omega\left[\frac{\partial u_1}{\partial x_1}\frac{\partial v_1}{\partial x_1}+\frac{\partial u_2}{\partial x_2}\frac{\partial v_2}{\partial x_2}+\frac{\nu}{1-\nu}\,\mathrm{div}u\ \mathrm{div}v\right.\\
&\left.+\frac{1}{2}\left(\frac{\partial u_1}{\partial x_2}+\frac{\partial u_2}{\partial x_1}\right)\left(\frac{\partial v_1}{\partial x_2}+\frac{\partial v_2}{\partial x_1}\right)\right]dx=\int_{\Gamma_N} g_{2,1}v_1+g_{2,2}v_2\,ds
\end{aligned}
$$

holds for all test functions $v \in V_0$.

Here $g_2 = (g_{2,1}, g_{2,2})^T$ denotes the surface tractions, E is Young's elasticity modulus, and ν is the Poisson ratio. The space V_0 is defined by $V_0 = \{v \in [H^1(\Omega)]^2 : v_1(x) = v_2(x) = 0 \text{ on } \Gamma_D\}$ and $\partial\Omega = \bar{\Gamma}_D \cup \bar{\Gamma}_N$, $\Gamma_D \cap \Gamma_N = \emptyset$. The domain Ω is given in Fig. 3 and Fig. 4 shows the triangulation $\mathcal{T}_4$ and the contour of the domain with its deformation.

Again we compare the CPU time needed for the generation of the FE systems in the nodal basis of piecewise linear functions and in the p-hierarchical basis.

Furthermore, we give results concerning the application of Algorithm 1 and its use as a preconditioner in the PCCG method. In the algorithms we use the same components as in the first example. We mention here that the constant in the strengthened Cauchy inequality (80) is relatively large, namely $\gamma = 0.94$ (see [12]), and therefore the convergence of Algorithm 1 is poor. For the $(l-1)$-grid method, which is used to solve the coarse-grid problem, we observed a convergence rate of 0.371. Therefore the bound on the convergence rate for Algorithm 1 cannot be better than 0.93.

In Tables 6 and 7 we summarize some results for Algorithm 1, Algorithm $1'(H, Q)$, and the MG(1)-PCCG method for the systems (87).

Finally, we compare the energy of the FE solutions obtained by the application of Algorithm 1 and Algorithm $1'(H, Q)$. From Table 8 we see that we have in both cases the same FE solution.

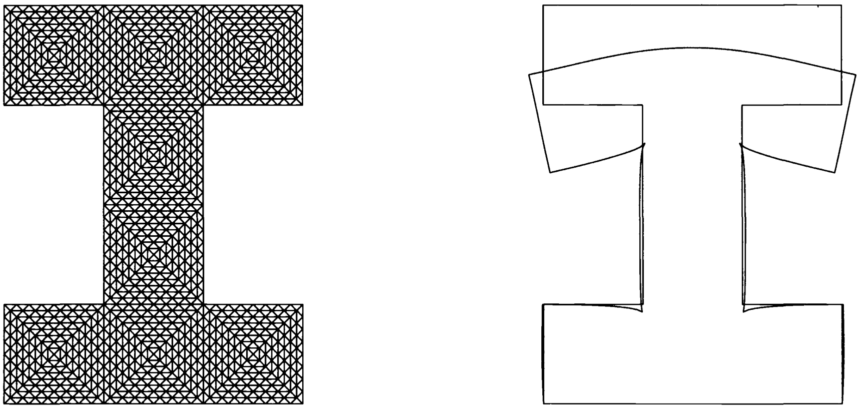


FIG. 4. *The triangulation (level* 4) *and the contour of the domain with its deformation.*

TABLE 6
Comparison of the CPU time needed for the generation of the FE systems.

Level l	Number of degrees of freedom	Number of triangles of $\mathcal{T}_{l-1}$	CPU time for the generation of	
			$K^{L,N}_{l-1}$, $K^{L,N}_{l}$, $\underline{f}^{L,N}_{l-1}$, $\underline{f}^{L,N}_{l}$	$K^{Q,H}_{l}$, $\underline{f}^{Q,H}_{l}$
3	586	128	0.27 sec	0.24 sec
4	2194	512	0.99 sec	0.96 sec
5	8492	2048	4.01 sec	3.89 sec

TABLE 7
Comparison of Algorithm 1, *Algorithm* $1'(H, Q)$, *and the MG*(1)*-PCCG method.*

l	Algorithm 1		Algorithm $1'(H, Q)$		MG(1)-PCCG method	
	Number of iterations	CPU time	Number of iterations	CPU time	Number of iterations	CPU time
3	25	3.51 sec	25	3.81 sec	9	1.32 sec
4	26	15.37 sec	26	16.88 sec	9	5.87 sec
5	25	63.11 sec	25	70.24 sec	9	23.40 sec

TABLE 8
Energy norm of the solution.

Level l	Energy norm of the solution obtained from	
	Algorithm 1	Algorithm $1'(H, Q)$
3	6.34388	6.34386
4	6.41933	6.41931
5	6.45374	6.45378

5. Conclusions. We have shown that multigrid τ-extrapolation can be interpreted as an implicit method to form higher order FE stiffness matrices. This is not only of theoretical interest, but leads to an efficient higher order multilevel solution technique for PDEs. In particular, this extrapolation technique can be used on unstructured meshes.

The resulting algorithm is competitive with multilevel methods that use higher order elements directly. The convergence rate and numerical work per iteration are comparable, but the algorithm has the advantage of a possibly simpler structure. In particular, the τ-extrapolation method is easy to incorporate into existing low-order methods, because it differs from the basic algorithm for linear elements only by a slight modification of the fine-to-coarse restriction process.

The alternative analysis for τ-extrapolation given in Rüde [20] is based on asymptotic expansions for quadrature rules over the triangle, and shows that the method can be generalized when the coefficients are not piecewise constant. In this case the linear combination of the stiffness matrices constitute an appropriate numerical quadrature formula for the quadratic stiffness matrix. This analysis also opens up the possibility of generalizing this technique for more than one extrapolation step. These methods would then be equivalent to polynomial finite elements of degree $p > 2$. Some preliminary results for these extensions are contained in Rüde [17], [18], [19].

Acknowledgment. We would like to thank the referees for many helpful comments.

REFERENCES

[1] O. Axelsson and I. Gustafsson, *Preconditioning and two–level multigrid methods of arbitrary degree of approximation*, Math. Comp., 40 (1983), pp. 219–242.

[2] R. E. Bank and T. F. Dupont, *An optimal order process for solving elliptic finite element equations*, Math. Comp., 36 (1981), pp. 35–51.

[3] K. Bernert *Tauextrapolation – theoretische Grundlagen, numerische Experimente und Anwendung auf die Navier–Stokes–Gleichungen*, Preprint 94_7, Preprint–Reihe der Chemnitzer DFG–Forschergruppe "Scientific Parallel Computing", Fakultät für Mathematik, Technische Universität Chemnitz–Zwickau, Germany, 1994.

[4] H. Blum, Q. Lin, and R. Rannacher, *Asymptotic error expansions and Richardson extrapolation for linear finite elements*, Numer. Math., 49 (1986), pp. 11–37.

[5] D. Braess, *The contraction number of a multigrid method for solving the Poisson equation*, Numer. Math., 37 (1981), pp. 387–404.

[6] A. Brandt, *Multigrid techniques:* 1984 *guide with applications to fluid dynamics*, GMD Studien, 85 (1984).

[7] W. Hackbusch, *Multi-Grid Methods and Applications*, Springer Series Comput. Math., Springer-Verlag, Berlin, 1985.

[8] W. Hackbusch and U. Trottenberg, eds., *Multigrid methods*, Lecture Notes in Mathematics, Springer-Verlag, Berlin, Heidelberg, New York, 1982; Proc. of the Conference held at Köln–Porz, November 23–27, 1981.

[9] M. Jung, *Konvergenzfaktoren von Mehrgitterverfahren für Probleme der ebenen linearen Elastizitätstheorie*, ZAMM, 67 (1987), pp. 165–173.

[10] ———, *Eine Einführung in die Theorie und Anwendung von Mehrgitterverfahren*, Wissenschaftliche Schriftenreihe 9, Technische Universität Karl–Marx–Stadt (Chemnitz–Zwickau) Germany, 1989.

[11] M. Jung, U. Langer, A. Meyer, W. Queck, and M. Schneider, *Multigrid preconditioners and their applications*, in Third Multigrid Seminar, Biesenthal, 1988, G. Telschow, ed., Report R–MATH–03/89, Karl–Weierstrass–Institut, Berlin, 1989, pp. 11–52.

[12] M. Jung, U. Langer, and U. Semmler, *Two-level hierarchically preconditioned conjugate gradient methods for solving linear elasticity finite element equations*, BIT, 29 (1989), pp. 748–768.

[13] M. Jung and U. Rüde, *Implicit extrapolation methods for multilevel finite element computations: Theory and application*, Preprint 94_11, Preprint–Reihe der Chemnitzer DFG–Forschergruppe "Scientific Parallel Computing", Fakultät für Mathematik, Technische Universität Chemnitz–Zwickau, Germany, 1994.

[14] J. F. Maitre and F. Musy, *The contraction number of a class of two-level methods, an exact evaluation for some finite element subspaces and model problems*, in Lecture Notes in Mathematics 960, Springer-Verlag, Berlin, 1982, pp. 535–544.

[15] G. Marchuk and V. Shaidurov, *Difference Methods and Their Extrapolations*, Springer-Verlag, New York, 1983.

[16] S. McCormick and U. Rüde, *On local refinement higher order methods for elliptic partial differential equations*, Internat. J. High Speed Comput., 2 (1990), pp. 311–334. Also available as Bericht I-9034, TU München, Germany.

[17] U. Rüde, *Multiple tau-extrapolation for multigrid methods*, Bericht I-8701, Institut für Informatik, TU München, Germany, January 1987.

[18] ———, *On the accurate computation of singular solutions of Laplace's and Poisson's equation*, in Multigrid Methods: Theory, Applications, Supercomputing: Proceedings of the Third Copper Mountain Conference on Multigrid Methods, April 5–10, 1987, S. McCormick, ed., Marcel Dekker, New York, 1988.

[19] U. RÜDE, *Extrapolation and related techniques for solving elliptic equations*, Bericht I-9135, Institut für Informatik, TU München, Germany, September 1991.

[20] ———, *Extrapolation techniques for constructing higher order finite element methods*, Bericht I-9304, Institut für Informatik, TU München, Germany, 1993.

[21] N. SCHIEWECK, *A multigrid convergence proof by a strengthened Cauchy inequality for symmetric elliptic boundary value problems*, in Second Multigrid Seminar, Garzau, November 5–8, 1985, G. Telschow, ed., Report R–Math–08/86, Karl–Weierstrass–Institut für Mathematik, Berlin, 1986, pp. 49–62.

[22] T. STEIDTEN AND M. JUNG, *Das Multigrid–Programmsystem FEMGPM zur Lösung elliptischer und parabolischer Differentialgleichungen einschließlich mechanisch–thermisch gekoppelter Probleme* (*Version 06.90*), Programmdokumentation, Sektion Mathematik, Technische Universität, Karl–Marx–Stadt (Chemnitz–Zwickau) 1990.

[23] C.-A. THOLE, *Beiträge zur Fourieranalyse von Mehrgittermethoden: V-cycle, ILU–Glättung, anisotrope Operatoren*, Diplomarbeit, Institut für Angewandte Mathematik, Universität Bonn, 1983.

[24] H. YSERENTANT, *On the multi-level splitting of finite element spaces*, Numer. Math., 49 (1986), pp. 379–412.

SIAM J. SCI. COMPUT.
Vol. 17, No. 1, pp. 180–192, January 1996

012

ON RED-BLACK SOR SMOOTHING IN MULTIGRID*

IRAD YAVNEH†

Abstract. Optimal relaxation parameters are obtained for red-black Gauss–Seidel relaxation in multigrid solvers of a family of elliptic equations. The resulting relaxation schemes are found to retain very high efficiency over an appreciable range of coefficients of the elliptic differential operator, yielding simple, inexpensive, and fully parallelizable smoothers in many situations where less cost-effective block- and alternating-direction schemes are commonly used.

Key words. multigrid, SOR, red-black Gauss–Seidel

AMS subject classifications. 65N12, 65N55

1. Introduction.

An important part of multigrid algorithms for the solution of discretized elliptic boundary-value problems is relaxation, whose purpose is to *smooth* the current error in the approximation, i.e., to efficiently reduce all error Fourier components that cannot be approximated on the coarser grids employed. For definite elliptic operators, the *smoothing factor* (defined below) normally provides an excellent prediction of the convergence factor that the two-grid cycle can achieve.

A well-known technique for improving the convergence properties of relaxation, when it is used as an iterative solver, is relaxation parameters (over- or under-relaxation). In the context of multigrid solution, where the main role of relaxation is to smooth the error rather than reduce it, it has been observed that such modifications are not cost-effective in many situations, particularly when applied to Gauss–Seidel relaxation (e.g., [1, §3.4], [5, §9.2]). (Of course, in other relaxation schemes a relaxation parameter may be essential for smoothing, such as the necessary damping in Jacobi relaxation.) For example, some improvement in the smoothing properties of Gauss–Seidel relaxation has been shown with slight over-relaxation in the case of the two-dimensional Laplace operator, but the two additional floating-point operations required rendered this modification ineffective [5].

It is a fundamental observation that any local relaxation process must lose its efficiency as the operator becomes anisotropic. Standard treatments of this problem are global methods, such as incomplete LU decomposition and line (or plane) relaxation or semi coarsening (or both), sometimes together with alternating-direction relaxation [2], [5], [7], [9], [10]. Frequently, the resulting solver is far more expensive (per relaxation sweep) than the usual solver which employs point relaxation and standard coarsening. This is especially true in parallel computation, where the more elaborate methods cannot always be implemented efficiently.

In making the analyses that led to the conclusions that (a) over-relaxation is inefficient, and (b) elaborate methods are necessary for anisotropic operators, a significant "middle ground" has often been neglected. Thus, while it is true that over-relaxation is not particularly helpful for the two-dimensional (2D) Poisson problem, and also that special measures are required when the ratio of coefficients differs from 1 by several orders of magnitude, it will be shown that relaxation parameters are quite useful when the anisotropy is moderate. This is also true in the case of isotropic operators in higher dimensions.

*Received by the editors May 20, 1994; accepted for publication (in revised form) December 5, 1994. This research was supported by the U.S.-Israel Binational Science Foundation, and also in part by the Loewengart Research Fund and by the Fund for the Promotion of Research at the Technion.

†Department of Computer Science, Technion—Israel Institute of Technology, Haifa 32000, Israel (irad@cs.technion.ac.il).

We consider scalar elliptic partial differential operators of the type

$$L = \sum_{i=1}^{n} c_i \partial_{x_i^2}^2 \,, \qquad \sum_{i=1}^{n} c_i = 1 \,, \tag{1}$$

on a rectangular periodic domain, where c_i $(i = 1, \dots, n)$ are real positive constants. The discretization of L is assumed to be the usual second-order accurate central differences involving only nearest neighbors (producing a $2n+1$-point "star"). Throughout this study, $n \geq 2$ is assumed. We denote the set of coordinate indices by $\mathcal{I} = \{1, \dots, n\}$, the position labels by a vector of integers $\mathbf{k} = (k_1, \dots, k_n)$, and the meshsize by h. A square mesh is assumed for simplicity, but rectangular-mesh problems are trivially reduced to equivalent square-mesh problems by replacing the coefficients c_i with $\tilde{c}_i = c_i (\frac{\tilde{h}}{h_i})^2$, where h_i is the meshsize corresponding to c_i, and $\tilde{h}$ is chosen such that $\sum_{i=1}^{n} \tilde{c}_i = 1$. For the square mesh,

$$x_i = hk_i \,, \qquad i \in \mathcal{I} \,.$$

We compute optimal relaxation parameters for Gauss–Seidel relaxation in red-black ordering. In §2 we analyze the case where only one relaxation sweep is performed between coarse-grid corrections. In §3 we consider the case of several relaxation sweeps. In §4 results of two-level analyses are obtained and compared to the smoothing-factor values, and numerical tests are reported for problems with varying coefficients and varying grid-spacing. The results are summarized and conclusions are drawn in §5.

2. Single relaxation sweep.

2.1. Smoothing analysis. "Pattern" relaxation methods, such as red-black Gauss–Seidel have been studied and analyzed extensively in the context of multigrid algorithms. (See especially [5], and also, e.g., [1], [3], [7], [8], and [10].) We describe the smoothing analysis here briefly.

It is a well-known observation for Gauss–Seidel relaxation of five-point star operators (in two dimensions) in red-black order, that two-dimensional subspaces of error Fourier components are invariant. This result immediately generalizes to $2n+1$-point star operators in n dimensions. Specifically, this relaxation couples each Fourier component $\exp(i\mathbf{k} \cdot \theta)$ only with $\exp(i\mathbf{k} \cdot \tilde{\theta})$, where

$$\theta = (\theta_1, \dots, \theta_n) \,, \qquad \tilde{\theta} = (\tilde{\theta}_1, \dots, \tilde{\theta}_n) \,, \tag{2}$$

$$-\pi < \theta_i \leq \pi \,, \tag{3}$$

and

$$\tilde{\theta}_i = \theta_i - \operatorname{sign}(\theta_i)\pi \,, \tag{4}$$

$i \in \mathcal{I}$. Here sign(0) is defined as -1. θ_i can really assume discrete values only, but, for convenience in obtaining h-independent upper bounds, below we assume that θ_i can take on any value in $(-\pi, \pi]$.

DEFINITION 1. *An element θ_i of mode θ is* smooth *if it satisfies*

$$-\frac{\pi}{2} < \theta_i \leq \frac{\pi}{2} \,.$$

Otherwise it is rough.

DEFINITION 2. *A Fourier mode θ is smooth if all its elements θ_i are smooth. Otherwise it is rough.*

We shall denote the set of smooth θ's by Θ_s and the set of rough θ's by Θ_r.

Fourier modes that are rough on the fine grid cannot be approximated on the twice-coarser grid that is assumed to provide the coarse-grid correction, since they alias with other components. Hence, they need to be treated efficiently by relaxation on the fine grid. Evidently, all pairs $(\theta, \tilde{\theta})$ consist of either two rough modes or one rough and one smooth mode. Without loss of generality, it will be our convention that $\tilde{\theta}$ is always rough.

The relaxation operator, $S(\theta, \tilde{\theta}) = \{s_{ij}\}$, is a two-by-two matrix which gives the amplitude of error Fourier components $(\exp(i\theta \cdot \mathbf{k}), \exp(i\tilde{\theta} \cdot \mathbf{k}))^T$ after one full relaxation sweep, when multiplied by their amplitudes before the sweep. It can be obtained by multiplying the operators of the two weighted-Jacobi half-sweeps performed over the black points (defined as gridpoints for which the sum of the indices k_i is odd) and the red points (whose index-sums are even). Now, a sweep over the black points amplifies components of black-point errors by the weighted-Jacobi symbol s_ω (see (6)–(7) below), without affecting the red-point errors, and vice versa. And since red-point and black-point error components can be expressed as sums and differences of the pairs of Fourier components, each half-sweep operator can be written as a two-by-two matrix. These are given by

$$\frac{1}{2}\begin{pmatrix} s_\omega(\theta)+1 & s_\omega(\tilde{\theta})-1 \\ s_\omega(\theta)-1 & s_\omega(\tilde{\theta})+1 \end{pmatrix} \quad \text{and} \quad \frac{1}{2}\begin{pmatrix} s_\omega(\theta)+1 & -s_\omega(\tilde{\theta})+1 \\ -s_\omega(\theta)+1 & s_\omega(\tilde{\theta})+1 \end{pmatrix}$$

for the red-point and black-point relaxation half-sweeps, respectively. Now, S is given by their product:

(5)
$$S = \frac{1}{4}\begin{pmatrix} [s_\omega(\theta)+1]^2 + [s_\omega(\tilde{\theta})-1][1-s_\omega(\theta)] & [s_\omega(\theta)+1][1-s_\omega(\tilde{\theta})] + s_\omega(\tilde{\theta})^2 - 1 \\ [s_\omega(\tilde{\theta})+1][1-s_\omega(\theta)] + s_\omega(\theta)^2 - 1 & [s_\omega(\tilde{\theta})+1]^2 + [s_\omega(\tilde{\theta})-1][1-s_\omega(\theta)] \end{pmatrix}.$$

Here, the weighted Jacobi relaxation symbols are given by

$$s_\omega(\theta) = 1 - \omega(1-C), \qquad s_\omega(\tilde{\theta}) = 1 - \omega(1+C), \tag{6}$$

where

$$C = \sum_{i=1}^{n} c_i \cos\theta_i, \tag{7}$$

and ω is the relaxation parameter. In the special case of $\omega = 1$, $s_1(\theta) = -s_1(\tilde{\theta}) = C$, and S is simplified appreciably.

In relaxation operators for which the Fourier components are eigenfunctions (such as damped Jacobi relaxation for the present problem), S is a scalar, and the smoothing factor μ is naturally defined as the largest absolute value of S over the space of rough modes. μ must give some information on the asymptotic reduction of rough error components by the relaxation, but the other parts of the multigrid cycle are idealized. This motivation led to the following highly successful extension to general S, introduced in [1] and [5] (see also [7], [8], and [10]): apply S and then annihilate the smooth modes, while leaving the rough modes unchanged, by projecting on the space of rough modes. When S is a two-by-two matrix, the projection operator Q, which acts as the idealized coarse-grid operator, is given by

$$Q = \begin{pmatrix} q(\theta) & 0 \\ 0 & q(\tilde{\theta}) \end{pmatrix}, \tag{8}$$

where $q = 0$ for a smooth argument and 1 otherwise. (Hence, our rough-$\tilde{\theta}$ convention implies $q(\tilde{\theta}) = 1$.) The smoothing factor μ, when a single relaxation sweep is performed between successive coarse-grid corrections, is defined by

$$\mu = \sup_h \max_\theta \rho(QS)\,, \tag{9}$$

where ρ denotes spectral radius. The implication of the *supremum* is that θ is allowed *any* value in $(-\pi, \pi]$. This will henceforth be implicitly assumed, and $\sup_h$ will be omitted for brevity.

Let

$$\mu^{(s)} = \max_{\theta\in\Theta_s} \rho(QS)$$

and

$$\mu^{(r)} = \max_{\theta\in\Theta_r} \rho(QS).$$

Note that $\mu = \max(\mu^{(s)}, \mu^{(r)})$. Our object is to find ω_{opt}, the relaxation parameter which minimizes μ, yielding μ_{opt}. In particular, μ_{opt} must obviously satisfy

$$\mu_{\text{opt}} \le \mu_{(\omega=1)} = (1 - c_{\min})^2 < 1\,, \tag{10}$$

where

$$0 < c_{\min} \overset{\text{def}}{=} \min_{i\in\mathcal{I}} c_i \le 0.5\,.$$

($\mu_{(\omega=1)}$ is derived in, e.g., [5] and [7] for $n = 2$ and in [10] for general n. The latter inequality is due to the fact that we are only considering $n \ge 2$.)

2.2. Derivation of ω_{opt}. For $\theta \in \Theta_s$, $q(\theta)$ vanishes as noted, leaving

$$\mu^{(s)} = \max_{\theta\in\Theta_s} |s_{22}|\,. \tag{11}$$

Now, $\theta \in \Theta_s$ implies by (7) that C can take on any value between 0 (when all components of θ equal $\frac{\pi}{2}$) and 1 (when all components of θ equal zero). Hence, (5), (6), and (11) yield after rearrangement

$$\mu^{(s)} = \frac{1}{2} \max_{C\in[0,1]} \left|\omega^2(C^2 + C) - 2\omega(1 + C) + 2\right|\,. \tag{12}$$

In order to find the maximum of the right-hand side of (12) over C in the relevant range, we must check the endpoints, $C = 0$ and $C = 1$, and the point at which the derivative with respect to C vanishes. Setting $C = 0$ yields $|\omega - 1|$. This implies, by (10), $0 < \omega_{\text{opt}} < 2$. Setting $C = 1$ yields $(\omega - 1)^2$, which is smaller than $|\omega - 1|$ in this range of ω. Equating the derivative with respect to C to zero yields after rearrangement $0.125|\omega^2 + 4\omega - 4|$, at $C = (2 - \omega)/2\omega$. This C is in [0, 1] so long as $2/3 \le \omega \le 2$. Indeed, it will be established below (Observation 4) that we are only interested in the range $1 \le \omega < 2$. For this range of ω we therefore have the following observation.

OBSERVATION 1. *For* $1 \le \omega < 2$,

$$\mu^{(s)} = 0.125(\omega^2 + 4\omega - 4)\,,$$

and $\mu^{(s)}$ *is consequently a monotonically increasing function of* ω *in* $[1, 2)$.

Proof.

$$0.125|\omega^2+4\omega-4|-|\omega-1|=0.125(\omega-2)^2>0\,. \qquad \square$$

For $\theta\in\Theta_r$, Q is the identity matrix, so $QS=S$, whose eigenvalues are given by

$$\lambda_{1,2}^{(r)}=\frac{1}{2}\left[s_{11}+s_{22}\pm\sqrt{(s_{11}-s_{22})^2+4s_{12}s_{21}}\right]. \tag{13}$$

Substitution from (5) and (6) yields

$$\begin{aligned} s_{11}+s_{22} &= \omega^2C^2-2(\omega-1)\,,\\ s_{11}-s_{22} &= \omega C(2-\omega)\,,\\ 4s_{12}s_{21} &= C^2\omega^4(C^2-1)\,, \end{aligned}$$

and hence

$$\lambda_{1,2}^{(r)}=\frac{1}{2}\left[\omega^2C^2-2(\omega-1)\pm\omega C\sqrt{\omega^2C^2-4(\omega-1)}\right]. \tag{14}$$

Now, $\theta\in\Theta_r$ implies that at least one of θ's components is rough (by Definition 2), but also that at least one of its components is smooth, else $\tilde{\theta}$ would be smooth, by (4), in contradiction to our convention. Hence, by (7), the relevant range of C in (14) is given by

$$|C|\le C_{\max}\overset{\text{def}}{=}1-c_{\min}\,. \tag{15}$$

Here, $C=C_{\max}$ is obtained when all of θ's components vanish, except that which corresponds to $c_{\min}$, which equals $-\frac{\pi}{2}$. Note that

$$0.5\le C_{\max}<1\,. \tag{16}$$

Note also in (14) that $\lambda_1(C)=\lambda_2(-C)$, so it suffices to consider nonnegative values of C, and this will be assumed henceforth for convenience.

OBSERVATION 2. *Denote*

$$D_\omega(C)=\omega^2C^2-4(\omega-1)\,, \tag{17}$$

and let $\bar{D}_\omega=D_\omega(C_{\max})$. Then,

$$\mu^{(r)}=|\omega-1|\,, \tag{18}$$

if $\bar{D}_\omega\le 0$, and

$$\mu^{(r)}=\frac{1}{2}\left[\omega^2C_{\max}^2-2(\omega-1)+\omega C_{\max}\sqrt{\omega^2C_{\max}^2-4(\omega-1)}\right], \tag{19}$$

if $\bar{D}_\omega\ge 0$.

Proof. If $\bar{D}_\omega<0$, then $\lambda_{1,2}^{(r)}$ are complex, and

$$|\lambda_1^{(r)}|=|\lambda_2^{(r)}|=\frac{1}{2}\left\{[\omega^2C^2-2(\omega-1)]^2-\omega^2C^2D_\omega(C)\right\}^{1/2}=|\omega-1|\,, \tag{20}$$

independently of C. And (20) clearly holds also when $\bar{D}_\omega=0$. On the other hand, whenever $D_\omega(C)\ge 0$ then, perforce, also $\omega^2C^2-2(\omega-1)\ge 0$. Hence, only the "+" sign in front of the square root in (14) needs to be considered when seeking the root that is larger in absolute value. Moreover, in this case the corresponding root is evidently a monotonically increasing function of C. By (15), this completes the proof. □

This yields the following observation.

OBSERVATION 3. *For $\bar{D}_\omega > 0$, $\mu^{(r)}$ is a monotonically decreasing function of ω.*

Proof. Differentiation of (19) with respect to ω yields after rearrangement

$$\frac{\partial \mu^{(r)}}{\partial \omega} = \frac{1}{\sqrt{\bar{D}_\omega}} \left[\sqrt{\bar{D}_\omega}(\omega C_{\max}^2 - 1) + C_{\max}(\bar{D}_\omega + \omega - 2) \right]. \tag{21}$$

Evidently, a necessary condition for this derivative to vanish is

$$\bar{D}_\omega(\omega C_{\max}^2 - 1)^2 = C_{\max}^2(\bar{D}_\omega + \omega - 2)^2,$$

which more algebra reduces to

$$(\omega - 1)(C_{\max}^2 - 1) = 0.$$

Now, substituting $C_{\max} = 1$ into (21) indeed makes the derivative vanish, but that is outside the range of $C_{\max}$, by (16). On the other hand, putting $\omega = 1$ yields for the derivative the negative value of $2(C_{\max}^2 - 1)$. Hence, the derivative is negative for all relevant values of $C_{\max}$. □

Now we have the following observation.

OBSERVATION 4.

$$1 \leq \omega_{\text{opt}} < 2.$$

That is, μ_{opt} is obtained by over-relaxation.

Proof. The obvious upper bound has already been established. To derive the lower bound we first note that when $\omega = 1$,

$$\mu^{(r)} = C_{\max}^2 \geq 0.25 > 0.125 = \mu^{(s)}.$$

(See (16) and Observation 1.) Hence, $\mu = \mu^{(r)}$ at $\omega = 1$, and the proof follows now from Observation 3 and the fact that $\bar{D}_\omega$ is positive for $\omega < 1$. □

We can now establish sharp bounds on ω_{opt} and μ_{opt}, by choosing ω such that $\bar{D}_\omega = 0$.

THEOREM 2.1.

$$\omega_{\text{opt}} < \omega_{ub} = \frac{2}{1 + \sqrt{1 - C_{\max}^2}},$$

and

$$\mu_{\text{opt}} < \mu_{ub} \stackrel{\text{def}}{=} \mu_{(\omega=\omega_{ub})} = 0.125(\omega_{ub}^2 + 4\omega_{ub} - 4) = \frac{1 + C_{\max}^2}{2\left(1 + \sqrt{1 - C_{\max}^2}\right)^2}.$$

Hence,

$$\mu_{\text{opt}} \leq 1 - \sqrt{8c_{\min}} + O(c_{\min}).$$

Proof. When $\omega = \omega_{ub}$, $\bar{D}_\omega = 0$ and $\mu^{(r)} = \omega_{ub} - 1 < \mu^{(s)}$ (see Observations 1 and 2). The upper bounds now follow from Observations 1 and 4. The final estimate is verified straightforwardly by substituting $1 - c_{\min}$ for $C_{\max}$, and finding that it applies to μ_{ub}. Indeed, this estimate can easily be shown to be an equality. □

COROLLARY 2.2. *$\bar{D}_\omega > 0$ when $\omega = \omega_{\text{opt}}$.*

Note that the upper-bound values of ω, which are shown below to be nearly optimal, are precisely of the form of the corresponding optimal values when SOR is used as a solver [6], but with $C^2_{\max}$ (the smoothing factor of red-black Gauss–Seidel) replacing the spectral radius of the Gauss–Seidel iteration matrix. Furthermore, it can easily be shown that

$$\frac{1 - \sqrt{1 - C^2_{\max}}}{1 + \sqrt{1 - C^2_{\max}}},$$

which is of the form of the optimal convergence factor for SOR when used as a solver (with the same substitution), is a lower bound on μ_{opt}.

Comparing μ_{opt} to $\mu_{(\omega=1)}$ in (10), we find that the former grows far more slowly as $c_{\min}$ decreases than the latter. This implies that employing over-relaxation allows point Gauss–Seidel relaxation in red-black ordering to retain high efficiency over a much larger range of coefficients c_i than the usual relaxation (see Fig. 2).

There seems to be no simple way to express ω_{opt} and μ_{opt} explicitly, although some implicit relations (whose simple proofs are omitted) are given below. However, Figs. 1 and 2 compare the optimal and upper-bound values of ω and μ, respectively, and demonstrate that the upper-bound values provide excellent approximations (see also below). This is also tested in numerical examples in §4.

Some implicit relations for ω_{opt} and μ_{opt} are

- ω_{opt} is the only $\omega \in [1, 2)$ which yields

  $$\mu^{(r)} = \mu^{(s)} (= \mu_{\text{opt}}),$$

  producing an equation for ω_{opt} in terms of $C_{\max}$, obtained from (19) and Observation 1.
- Let ω_{ub} and μ_{ub} be the upper-bound values corresponding to some particular $C^2_{\max}$. Then they are equal to the optimal values, ω_{opt} and μ_{opt}, which correspond to

  $$\tilde{C}^2_{\max} = C^2_{\max} + \frac{(\omega_{ub} - 2)^4}{64\omega^2_{ub}\mu_{ub}}.$$

  The smallness of the last term (especially as $C_{\max}$ tends to 1, and ω_{ub} to 2) plus the moderate sensitivity of ω_{opt} (and μ_{opt}) to variations in $C_{\max}$, which is verified by differentiation of the above-mentioned implicit equation, imply that the upper-bound values are close to the optimal values. This derivation is omitted, as the result is more clearly evident in Figs. 1 and 2.

3. Several relaxation sweeps. Unlike methods of relaxation for which Fourier components are eigenfunctions, the smoothing factor of red-black Gauss–Seidel relaxation depends on the number of sweeps performed between successive coarse-grid corrections. Let ν denote the number of sweeps. Then the smoothing factor is naturally defined by [1] and [5]

$$\mu = \left[\sup_h \max_\theta \rho(QS^\nu) \right]^{1/\nu}. \tag{22}$$

Since, for $\theta \in \Theta_r$, Q is the identity matrix, we have the following observation.

OBSERVATION 5. *$\mu^{(r)}$ is independent of ν.*

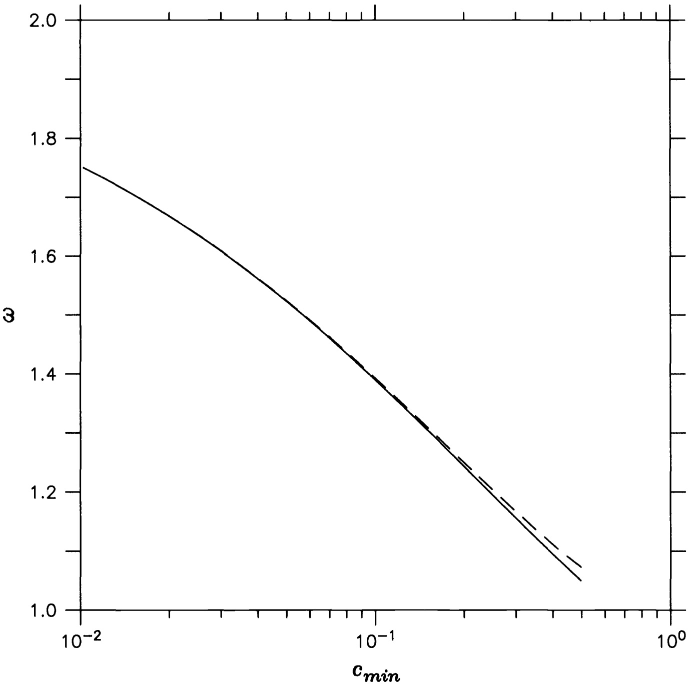


FIG. 1. ω_{opt} *(solid) and* ω_{ub} *(dashes) as a function of* $c_{\min}$.

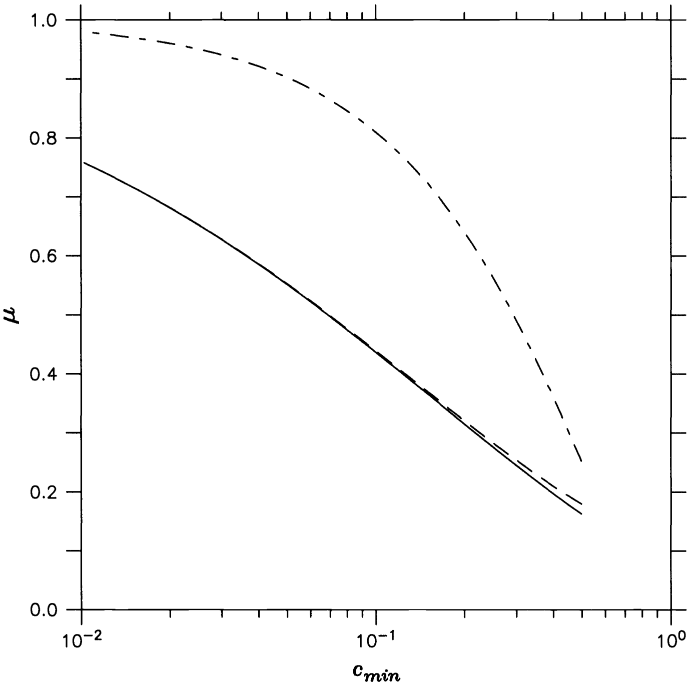


FIG. 2. μ_{opt} *(solid),* μ_{ub} *(dashes) and* $\mu_{(\omega=1)}$ *(alternating dashes) as a function of* $c_{\min}$.

Now denote

$$S^{\nu} = \{s_{ij}^{(\nu)}\}.$$

Then, we have the following observation.

OBSERVATION 6.

$$\mu^{(s)} = \max_{C \in [0,1]} |s_{22}^{(\nu)}|. \tag{23}$$

As ν is increased, $s_{22}^{(\nu)}$ has more and more terms, and we resort to numerical calculation in order to obtain $\mu^{(s)}$, and consequently μ. Figures 3 and 4 show ω_{opt} and μ_{opt} for $\nu = 1, 2, 3$. It is seen that there is some deterioration in performance (per relaxation sweep) when the number of sweeps performed between coarse-grid corrections is increased (as is generally the case with no over-relaxation too). However, this effect diminishes as $c_{\min}$ becomes small. Also, optimal ω is smaller for larger ν, but it is demonstrated below that this effect should be ignored in practical use, and ω_{ub} given in Theorem 2.1 is a very good working value.

4. Two-level analyses and numerical tests. The predictions of the smoothing analysis were compared with two-level analyses (which here predict the two-level performance exactly), and with results of numerical calculations. In all the examples, bi-linear interpolation was used for prolongation and its adjoint full-weighting operator for residual-restriction, with vertex-centered discretization and standard coarsening. Bi-cubic interpolation was also tested, with results that were usually slightly better only in the more isotropic cases, and somewhat worse otherwise.

Example 1. Analyses with the full two-level operator (two-level analyses, e.g., [1, §4.1], [10]) were carried out for the two-dimensional problem with several values of $c_{\min}$. The results are summarized in Table 1, which shows two-level convergence factors per relaxation sweep, denoted μ^{tl}. The values of ω compared are ω_{ub} from Theorem 2.1, ω_{opt} (Fig. 3), and ω_{best}, which denotes the over-relaxation parameter that yielded the best two-level results obtained. The sixth column shows the prediction of the smoothing analysis, using ω_{opt} (Fig. 4), and the seventh column shows the usual $\omega = 1$ performance. It is seen that there are only very minor differences between the results with the different near-optimal over-relaxation parameters, and that ω_{ub} is a very good working value. Indeed, in some cases it yields even better results than the optimum predicted by the smoothing analysis. In all cases over-relaxation yielded a large improvement in performance, but of course the relative gain (measured by the ratios of $\log \mu^{tl}$) is greater as $c_{\min}$ is smaller. The last two columns show the asymptotic error-reduction factor per cycle obtained by V cycles with two prerelaxation sweeps and one postrelaxation sweep per level for the problem

$$u_{xx} + u_{yy} + \epsilon u_{zz} = 0,$$

on a triply-periodic cube. Here, $\epsilon = 2c_{\min}/(1 - c_{\min})$. The finest resolution was 64^3 and five levels were employed. The initial field was random with uniform distribution. The right-hand side was set to 0, and 100 cycles were performed, renormalizing the solution after every cycle so as to avoid roundoff errors. (Of course, due to the linearity of the problem, the asymptotic convergence factors are independent of the choice of right-hand side function.) An appreciable improvement in performance is obtained by using the over-relaxation parameter ω_{ub}, which becomes more marked for smaller $C_{\min}$. For example, convergence using ω_{ub} is about twice as fast for $C_{\min} = 0.2$ and more than two and a half times as fast for $C_{\min} = 0.1$.

Example 2. V(1,1) cycles were implemented for the Poisson problem in three dimensions with the same specifications as in Example 1. Asymptotic error-reduction factors per cycle

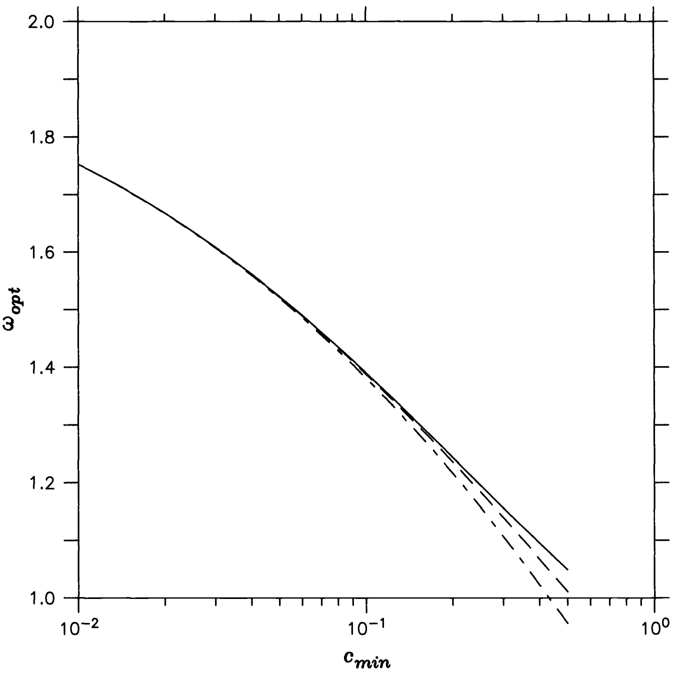


FIG. 3. *ω_{opt} as a function of c_{min}, with one (solid), two (dashes), and three (alternating dashes) relaxation sweeps ν between successive coarse-grid corrections.*

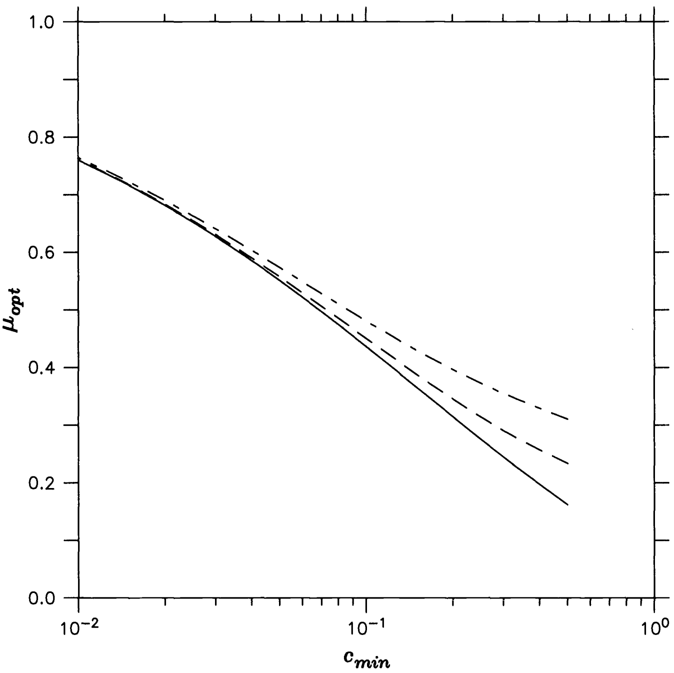


FIG. 4. *μ_{opt} as a function of c_{min}, with one (solid), two (dashes), and three (alternating dashes) relaxation sweeps ν between successive coarse-grid corrections.*

TABLE 1

Comparison of two-level analysis error-reduction factors (per relaxation sweep) using several values of over-relaxation parameter ω: ω_{ub} of Theorem 2.1, ω_{opt} of Fig. 3, and ω_{best}—the over-relaxation parameter which yielded the best two-level results. In the sixth column is the smoothing-analysis prediction of Fig. 4, and the seventh column shows the usual $\omega = 1$ performance. ν denotes the number of relaxation sweeps performed between successive coarse-grid corrections. The last two columns show the error-reduction factor per cycle obtained by V(2,1) cycles using $\omega = \omega_{ub}$ and $\omega = 1$, respectively (see text for details).

$c_{\min}$	ν	$\mu^{tl}_{\omega_{ub}}$	$\mu^{tl}_{\omega_{\mathrm{opt}}}$	$\mu^{tl}_{\omega_{\mathrm{best}}}$	μ_{opt}	μ^{tl}_{1}	$V(2,1)_{\omega_{ub}}$	$V(2,1)_1$
$\frac{1}{3}$	1	0.236	0.226	0.226	0.227	0.444	0.086	0.150
	2	0.272	0.277	0.262	0.277	0.444		
	3	0.315	0.361	0.302	0.345	0.444		
$\frac{1}{5}$	1	0.313	0.316	0.308	0.315	0.640	0.088	0.293
	2	0.354	0.347	0.345	0.345	0.640		
	3	0.378	0.397	0.373	0.397	0.640		
$\frac{1}{10}$	1	0.426	0.437	0.425	0.437	0.810	0.222	0.552
	2	0.445	0.450	0.442	0.451	0.810		
	3	0.487	0.481	0.480	0.481	0.810		

TABLE 2

Asymptotic error-reduction factors (E.R.F.) per V(1,1) cycle for the Poisson problem in three dimensions. In comparison, a V(2,1) cycle with $\omega = 1$ yields an error-reduction factor of 0.150 (see Table 1).

ω	E.R.F
1.	0.240
1.146 (ω_{ub})	0.128
1.114 (ω_{opt})	0.147
1.164 (ω_{best})	0.113

are compared in Table 2 for over-relaxation parameters 1, ω_{ub}, ω_{opt}, and ω_{best}. Even in this fully isotropic problem over-relaxation yields a significant improvement. ω_{ub} is slightly better than ω_{opt}, as ω_{best} is somewhat larger than predicted (as was the case for $c_{\min} = 1/3$ shown in Table 1). In comparison, a V(2,1) cycle with $\omega = 1$ (which is of course more expensive then the over-relaxed V(1,1) cycles even in the case of the simple Poisson operator) yielded an error-reduction factor 0.150 (see Table 1)—slightly worse than the 0.128 factor obtained with ω_{ub} which appears in Table 2.

Example 3. Since smoothing here is a local process, it seems natural to vary the over-relaxation parameter in space as a function of $c_{\min}$, so as to maintain locally-optimal smoothing throughout the domain. We tested this approach in this and the next examples. V(2,1) and V(2,2) cycles were implemented for the variable-coefficient two-dimensional ($n = 2$) problem

$$(1 + \epsilon \sin x)u_{xx} + (1 + \epsilon \sin y)u_{yy} = 0\,, \tag{24}$$

and three-dimensional ($n = 3$) problem

$$(1 + \epsilon \sin x)u_{xx} + (1 + \epsilon \sin y)u_{yy} + (1 + \epsilon \sin z)u_{zz} = 0\,. \tag{25}$$

The domain of solution was a square (cube) of side 2π. In the three-dimensional problem the same specifications as in Examples 1 and 2 were used. In the two-dimensional problem we solved both with periodic and Dirichlet boundary conditions. The latter were homogeneous, to avoid roundoff errors, and again 100 cycles were implemented. The finest resolution in the two-dimensional problem was 128^2 (127^2 not including boundary points in the Dirichlet problem), and six levels were employed. The error-reduction factors per cycle are shown

TABLE 3
Error reduction factors of V(2,1) cycles.

	Error Reduction Factors					
	$n=2$				$n=3$	
	Dirichlet		Periodic		Periodic	
ϵ	$\omega=\omega_{ub}$	$\omega=1$	$\omega=\omega_{ub}$	$\omega=1$	$\omega=\omega_{ub}$	$\omega=1$
0.50	0.053	0.190	0.057	0.191	0.092	0.369
0.80	0.159	0.513	0.167	0.513	0.236	0.675
0.90	0.297	0.712	0.307	0.712	0.351	0.810
0.95	0.443	0.837	0.458	0.837	0.443	0.887

TABLE 4
Error reduction factors of V(2,2) cycles.

	Error Reduction Factors					
	$n=2$				$n=3$	
	Dirichlet		Periodic		Periodic	
ϵ	$\omega=\omega_{ub}$	$\omega=1$	$\omega=\omega_{ub}$	$\omega=1$	$\omega=\omega_{ub}$	$\omega=1$
0.50	0.041	0.120	0.039	0.120	0.056	0.267
0.80	0.075	0.410	0.075	0.410	0.112	0.592
0.90	0.155	0.634	0.165	0.634	0.193	0.755
0.95	0.358	0.788	0.279	0.788	0.273	0.852

in Tables 3 and 4 for various values of ϵ, which determine the anisotropy, with $\omega = 1$ and $\omega = \omega_{ub}$ locally. Evidently, over-relaxation improves the usability of point relaxation very significantly. Indeed, in three dimensions, even moderate anisotropy necessitates over-relaxation for obtaining a useful algorithm.

Example 4. In a commonly used discretization for planetary flow simulations on the sphere, the gridlines lie along lines of latitude and longitude, and the grid is thus anisotropic near the poles. If the meshsizes in a two-dimensional problem are chosen to be equal at the equator, then their ratio elsewhere is approximately $\cos\phi$, where ϕ is the latitude (ranging from 0 at the equator to $\pm\pi/2$ at the poles). For the two-dimensional Poisson equation, this implies an equivalent anisotropy of approximately

$$c_{\min} \approx \frac{\cos^2\phi}{1+\cos^2\phi}\,.$$

One can apply more relaxation sweeps in regions that are far from the equator, but at some stage line relaxation (along latitude lines) is used. Over-relaxation can be employed to reduce the region where line relaxation needs to be used, while also reducing the number of sweeps that need to be performed where point relaxation is used. A particular example occurs in the shallow-water equations, especially when semi-Lagrangian discretization is used. The principal part of the matrix operator is then normally composed of three Laplace operators. In [4], over-relaxation is introduced for this problem, with the result that the total work spent on line relaxation is reduced to about 1/3, while the total work spent on point relaxation remains about the same (with less sweeps per latitude-line on the average, but more lines to sweep over). Here, V(ν, ν) cycles are employed, where ν is varied according to latitude, given some desired convergence factor. Line relaxation is used (with $\nu = 1$) whenever point relaxation would require $\nu > 4$ for the desired convergence rate. The relative gain quoted is not sensitive to the required convergence factor. These tests were performed at a resolution of 128 by 65 points along lines of latitude and longitude, respectively.

5. Concluding remarks. Optimal over-relaxation parameters for red-black Gauss–Seidel in multigrid algorithms were calculated for (1), and an upper bound, which was shown to provide a good working value, was established in Theorem 2.1. Employing over-relaxation is always more cost-effective than the usual red-black Gauss–Seidel in three dimensions or higher. In two dimensions over-relaxation is also better, unless the operator is very isotropic.

The extra cost required for over-relaxing, assuming that the optimal parameters are precomputed and stored, is about one or at most two operations per gridpoint. This may not be negligible in the case of the Laplace operator in two dimensions, but is quite unimportant in almost any relevant "real" problem.

The relative gain in using over-relaxation, compared to the usual relaxation, grows rapidly as the problem becomes more anisotropic. But of course the overall efficiency of the solver then decreases even when over-relaxation is used. Hence, full robustness can only be achieved by more elaborate global techniques, which are also more expensive (sometimes *much* more expensive on parallel machines). However, when the anisotropy is known to be moderate (perhaps only in part of the domain, as in Example 4—solution on a sphere), SOR should be employed. An example of this is large-scale, stably stratified planetary flows (e.g., quasi-geostrophic equations), where $c_{\min}$ will usually be about an order of magnitude smaller than 1 at worst.

Varying the relaxation parameter in space as a function of the anisotropy was found to perform well, especially if the over-relaxation parameters are precomputed and stored. One can then also use the prediction of μ_{ub} to choose the number of sweeps performed in various regions, as was done in Example 4 [4].

Acknowledgment. The author would like to thank John Ruge for his valuable contribution, and Bruno Welfert for pointing out the similarity of the present upper-bound over-relaxation parameter to the classical value.

REFERENCES

[1] A. Brandt, *Guide to multigrid development*, in Multigrid Methods, W. Hackbusch and U. Trottenberg, eds., Springer-Verlag, Berlin, 1982, pp. 220–312.

[2] R. Kettler, *Analysis and comparison of relaxation schemes in robust multigrid and conjugate gradient methods*, in Multigrid Methods, W. Hackbusch and U. Trottenberg, eds., Springer-Verlag, Berlin, 1982, pp. 502–534.

[3] C.-C. J. Kuo and B. C. Levy, *Two-color Fourier analysis of the multigrid method with red-black Gauss–Seidel smoothing*, Appl. Math. Comput., 29 (1989), pp. 69–87.

[4] J. Ruge, Y. Li, S. F. McCormick, A. Brandt, and J. R. Bates, *A nonlinear multigrid solver for a semi-Lagrangian potential vorticity-based barotropic model on the sphere*, SIAM J. Sci. Comput., submitted.

[5] K. Stüben and U. Trottenberg, *Multigrid methods: Fundamental algorithms, model problem analysis and applications*, in Multigrid Methods, W. Hackbusch and U. Trottenberg, eds., Springer-Verlag, Berlin, 1982, pp. 1–176.

[6] R. S. Varga, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1962.

[7] P. Wesseling, *An Introduction to Multigrid Methods*, Pure and Applied Mathematics, John Wiley & Sons, Chichester, 1992.

[8] ———, *A survey of Fourier smoothing analysis results*, in Multigrid Methods III, W. Hackbusch and U. Trottenberg, eds., Birkhäuser-Verlag, Basel, 1991, pp. 105–127.

[9] ———, *A robust and efficient multigrid method*, in Multigrid Methods, W. Hackbusch and U. Trottenberg, eds., Springer-Verlag, Berlin, 1982, pp. 614–630.

[10] I. Yavneh, *Multigrid smoothing factors for red-black Gauss–Seidel applied to a class of elliptic operators*, SIAM J. Numer. Anal., 32 (1995), pp. 1126–1138.

SIAM J. SCI. COMPUT.
Vol. 17, No. 1, pp. 193–216, January 1996

MULTILEVEL IMAGE RECONSTRUCTION WITH NATURAL PIXELS*

VAN EMDEN HENSON†, MARK A. LIMBER‡, STEPHEN F. MCCORMICK§, AND BRUCE T. ROBINSON¶

Abstract. The sampled Radon transform of a two-dimensional (2D) function can be represented as a continuous linear map $A : L_2(\Omega) \to \mathbf{R}^N$, where $(Au)_j = \langle u, \psi_j \rangle$ and ψ_j is the characteristic function of a strip through Ω approximating the set of line integrals in the sample. The image reconstruction problem is: given a vector $\mathbf{b} \in \mathbf{R}^N$, find an image (or density function) $u(x, y)$ such that $Au = \mathbf{b}$. In general there are infinitely many solutions; we seek the solution with minimal 2-norm, which leads to a matrix equation $B\mathbf{w} = \mathbf{b}$, where B is a square dense matrix with several convenient properties. We analyze the use of Gauss–Seidel iteration applied to the problem, observing that while the iteration formally converges, there exists a *near null space* into which the error vectors migrate, after which the iteration stalls. The null space and near null space of B are characterized in order to develop a multilevel scheme. Based on the principles of the multilevel projection method (PML), this scheme leads to somewhat improved performance. Its primary utility, however, is that it facilitates the development of a PML-based method for *spotlight tomography*, that is, local grid refinement over a portion of the image in which features of interest can be resolved at finer scale than is possible globally.

Key words. tomography, Radon transform, multigrid

AMS subject classifications. 92C55, 44A12, 65R10, 65N55

1. Introduction.

In this paper, we consider a model of transmission and emission tomography and an associated image reconstruction technique. The reconstruction technique approximates a minimum norm solution to an underdetermined linear inversion problem, based on an infinite-dimensional formulation of the tomographic inversion problem. This formulation of the problem avoids the traditional square pixel discretization of the image space and leads to a smaller, but dense, matrix problem (compared to traditional algebraic reconstruction techniques). This approach leads to what have been termed "natural pixels" in [1], and the "optimal grid" in [10].

Following the development of the natural pixel discretization, we consider solution techniques for the resulting linear system. In particular, we employ Gauss–Seidel iteration, analyze its performance, and then introduce a multilevel projection method (PML) for accelerating convergence.

2. Image reconstruction and the Radon transform.

We formulate the image reconstruction from projection problems in a general setting, but concentrate on a parallel beam geometry, for which we have implemented our ideas. The basic idea in tomography is that an object is subjected to a dose of radiation, either by passing X-rays through the object, or (if the object is a living patient) by administering a radiopharmaceutical. The amount of radiation leaving the object can be measured, compared with the original amount, and the difference is a measurement of the attenuation (transmission tomography) or activity (emission tomography) within the object. In parallel beam geometry, the data is collected in collimated bins, so that any activity detected in a particular bin can be attributed to the strip emanating perpendicularly out of the detector, with width equal to that of the bin.

To model this apparatus, let $u(x, y)$ be a function of the spatial variables x and y describing the activity in the object. Typically this is some physical quantity, such as the material density

*Received by the editors June 8, 1994; accepted for publication (in revised form) January 25, 1995.

†Department of Mathematics, Naval Postgraduate School, Monterey, CA 93943 (vhenson@math.nps.navy.mil).

‡Auto-trol Technology Corporation, Denver, CO 80231 (marlim@Auto-trol.com).

§Program in Applied Mathematics, University of Colorado, Boulder, CO 80309 (stevem@newton.colorado.edu).

¶Accurate Information Systems, Eatontown, NJ 07724 (brobinso@Accurate.com).

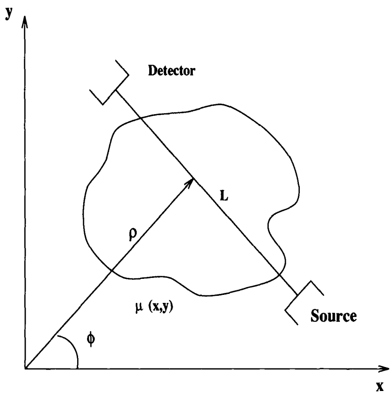


FIG. 1. *The geometry of the Radon transform.*

of the subject. The vector **f** represents the projection data. The data acquisition is modeled by

$$Au = \begin{pmatrix} \int_{\mathbf{R}^2} u(x,y)\psi_1(x,y)dxdy \\ \vdots \\ \int_{\mathbf{R}^2} u(x,y)\psi_N(x,y)dxdy \end{pmatrix} = \begin{pmatrix} f_1 \\ \vdots \\ f_N \end{pmatrix} = \mathbf{f}, \tag{1}$$

where the function ψ_k is the characteristic function of the kth strip through the image, within which passes (or emanates) the energy collected by the kth detector. For this to be well defined, we restrict our function space to be $L_2(\Omega)$, where Ω is a compact subset of $\mathbf{R}^2$, called the *image space*. Thus,

$$(Au)_j = \langle \psi_j, u \rangle$$

defines a continuous linear map $A : L_2(\Omega) \to \mathbf{R}^N$. The basic problem of computer assisted tomography is to reconstruct the image $u(x, y)$ from a collection of measured strip integrals, collected at various angles. When this problem can be solved, it is done through some approximate inversion of the Radon transform, which is defined as follows.

Let $u(x, y)$ be a function defined on the region $\Omega \in \mathbf{R}^2$. Letting L denote any line in $\mathbf{R}^2$, the set of line integrals of $u(x, y)$, along all possible lines L, is a function of two variables, and is known as the *Radon transform* of $u(x, y)$, provided the integral exists. Formally,

$$[Ru](\rho, \phi) = \int_L u(x,y)ds = \int u(x,y)\delta(x\cos\phi + y\sin\phi - \rho)dxdy,$$

where δ is the Dirac delta function. The line L is parametrized by

$$\rho = x\cos\phi + y\sin\phi, \tag{2}$$

where ρ is the signed distance from the origin and ϕ is an angle measured counter-clockwise from the positive x-axis. Thus, (2) determines the equation of a line in the xy-plane normal to the unit vector $\vec{\xi} = (\cos\phi, \sin\phi)^T$. Figure 1 shows the geometry of the Radon transform of a function $u(x, y)$ in terms of this parameterization.

Viewing the Radon transform as an operator, the image reconstruction problem can be cast as $Ru = f$, where f represents the collection of measured line integrals. Given **f**, a finite sampling of f, we model the problem $Ru = f$ with $Au = \mathbf{f}$, since each of the strip integrals $\langle \psi_j, u \rangle$ approximates a collection of line integrals, for those lines falling within the strip. Hence, the data **f** forms a sampling of the continuous Radon transform. We will refer to the set of strip integrals (1) as the *strip averaged* Radon transform.

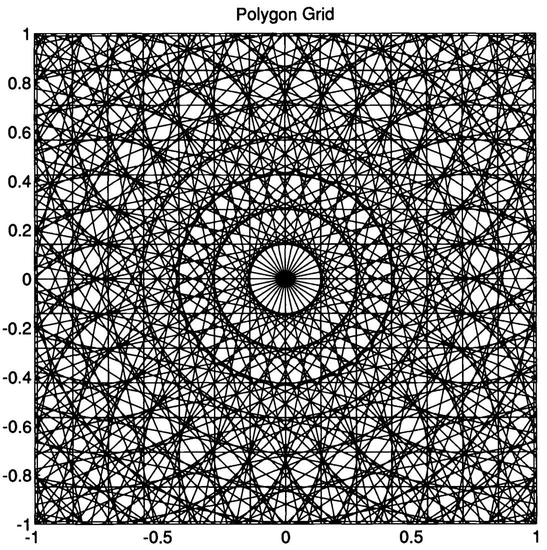


FIG. 2. *A representative grid of polygons. The optimal solution is constant on each polygon. This particular optimal grid corresponds to having* 16 *detector bins of uniform width for each of* 20 *angles, taken at regular angular intervals between* 0 *and* π.

3. Optimal grid discretization. Suppose there are M angles ϕ_j for $j = 1 : M$, such that $\phi_1 = 0 < \phi_2 < \phi_3 < \cdots < \phi_M < \pi$, and that at each angle ϕ_j there are $n(j)$ strips, or detector bins. Then $N = \sum_{j=1}^{M} n(j)$ gives the total number of data points. Suppose the image space Ω is some convex, compact region in $\mathbf{R}^2$, and assume that for the jth angle the $n(j)$ strips are parallel, nonoverlapping, and entirely cover Ω. Let $\psi_\ell(x, y)$ be the characteristic function of the ℓth strip. Then the discrete strip averaged Radon transform is the map $A : L_2(\Omega) \to \mathbf{R}^N$ defined in equation (1).

Assuming the system $Au = \mathbf{f}$ is consistent, it is underdetermined; that is, since A is a linear mapping from an infinite-dimensional space $L_2(\Omega)$ to the finite-dimensional space $\mathbf{R}^N$, the null space of A, $NS(A)$, is infinite-dimensional. If there are any solutions to $Au = \mathbf{f}$, there are infinitely many. We must select some representative solution image u from the infinite number of feasible images. The minimum norm solution to the equation $Au = \mathbf{f}$ is given by $u(x, y) = A^*\mathbf{w}$, where $\mathbf{w}$ solves the $N \times N$ system

$$AA^*\mathbf{w} = \mathbf{f}.$$

We write this system as $B\mathbf{w} = \mathbf{f}$ and concentrate on efficient methods to solve it. Note that forming $A^*\mathbf{w}$ corresponds to *backprojecting* the vector $\mathbf{w}$ over the image space.

A simple formula can be used to construct the matrix B. Note that $A^* : \mathbf{R}^N \to L_2(\Omega)$ is defined by

$$[A^*\mathbf{w}](x, y) = \sum_{i=1}^{N} w_i \psi_i(x, y). \tag{3}$$

Since the ψ_i are characteristic functions, we observe from equation (3) that the optimal image u is piecewise constant on the set of polygons defined by the intersections of the strips, at all angles. This set of polygons we term the *optimal grid*, as shown in Fig. 2.

The (i, j)th entry of B can be determined by computing the jth entry of $B\mathbf{e}_i$ where $\mathbf{e}_i$ is the ith standard basis vector in $\mathbf{R}^N$. Specifically,

$$(B\mathbf{e}_i)_j = \left(AA^*\mathbf{e}_i\right)_j = (A\psi_i(x, y))_j = \int_\Omega \psi_i(x, y)\psi_j(x, y)dxdy = \langle\psi_i, \psi_j\rangle.$$

Thus, B is an $N \times N$ matrix with entries

$$b_{ij} = \langle\psi_i, \psi_j\rangle. \tag{4}$$

We immediately have the following theorem.

THEOREM 1. *The $N \times N$ matrix B is nonnegative, symmetric, positive semidefinite with positive diagonal entries, $b_{ii} > 0$.*

Proof. This follows immediately since $\psi_k \geq 0$, $B = AA^*$, $\langle u, AA^*u\rangle = \langle A^*u, A^*u\rangle$, and $b_{ii} = \langle\psi_i, \psi_i\rangle$. □

Since we will employ iterative methods, it is important to identify those vectors that may cause difficulty in the iteration process. That is, we seek to characterize the eigenvectors of $\mathbf{R}^N$ that are associated with small nonzero eigenvalues of B. Such eigenvectors have the property that $B\mathbf{w}$ is small (in norm) compared to $\mathbf{w}$, and error vectors of this nature have residuals that are small compared to the error. We refer to them as vectors in the *near null space*, and assert that their presence in the error causes slow convergence. The near null space will be studied in §4.2. For now, we are concerned with characterizing the null space of B, which is just the null space of AA^*:

$$NS(B) = NS(AA^*) = NS(A^*). \tag{5}$$

For this characterization we need the following definition.

DEFINITION. *A vector* $\mathbf{w} \in \mathbf{R}^N$ *is said to be* constant by angle *if, upon writing*

$$\mathbf{w} = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_M \end{bmatrix}, \qquad \mathbf{v}_j \in \mathbf{R}^{n(j)}, \qquad \mathbf{v}_j = \begin{bmatrix} \alpha_1^j, \\ \alpha_2^j, \\ \vdots \\ \alpha_{n(j)}^j \end{bmatrix},$$

then $\alpha_i^j = \alpha_{i+1}^j$, $i = 1, \ldots, n(j) - 1$, *for* $j = 1 : M$.

Using this definition, we may show the following.

THEOREM 2. *A vector* $\mathbf{w} \in \mathbf{R}^N$ *is in* $NS(B)$ *if and only if* $\mathbf{w}$ *is constant by angle and the elements of* $\mathbf{w}$ *sum to zero*; *that is,* $\sum_{i=1}^N w_i = 0$.

Proof. Let $\{\hat\psi_k(x, y) | k = 1 : M\}$ be the set of strips that contain the point (x, y). There is one such strip from each projection angle. Let $\{\hat w_k | k = 1 : M\}$ be the set of coefficients associated with those strips. Clearly, if $\mathbf{w}$ is constant by angle and sums to zero, it is in $NS(B)$, since

$$[A^*\mathbf{w}](x, y) = \sum_{i=1}^N w_i\psi_i(x, y) = \sum_{k=1}^M \hat w_k\hat\psi_k(x, y) = 0.$$

Conversely, suppose $\mathbf{w} \in NS(A^*)$ so that

$$\sum_{i=1}^N w_i\psi_i(x, y) = 0 \quad \forall(x, y) \in \Omega.$$

Writing $\mathbf{w}$ as in the definition of constant by angle, that is,

$$\mathbf{w} = [\mathbf{v}_1^T, \ldots, \mathbf{v}_M^T]^T, \quad \mathbf{v}_j \in \mathbf{R}^{n(j)}, \quad \mathbf{v}_j = [\alpha_1^j, \ldots, \alpha_{n(j)}^j]^T,$$

then the objective is to show that for the jth projection angle, the subvector $\mathbf{v}_j \in \mathbf{R}^{n(j)}$ is constant. Without loss of generality, we show only that $\mathbf{v}_1$ is constant.

Denote the jth strip at the first angle as Ω_j. Consider the partitioning of the image space Ω into the set of polygons determined by the intersections of all strips at all angles *except the first.* Clearly, for each j, the boundary between Ω_j and Ω_{j+1} intersects the interior of at least one of these polygons. Hence, it is possible to select two points, $(x_1, y_1) \in \Omega_j$ and $(x_2, y_2) \in \Omega_{j+1}$, such that the line segment joining (x_1, y_1) and (x_2, y_2) lies entirely in one strip emanating from each of the other angles. That is, the line segment lies entirely in each of the strips $\Omega_{k_2}, \ldots, \Omega_{k_M}$.

By construction,

$$\begin{aligned} 0 = [A^*\mathbf{w}](x_1, y_1) - [A^*\mathbf{w}](x_2, y_2) &= \left(\alpha_j^1 - \alpha_{j+1}^1\right) + \sum_{\ell=2}^{M} w_{k_\ell} \left(\psi_{k_\ell}(x_1, y_1) - \psi_{k_\ell}(x_2, y_2)\right) \\ &= \alpha_j^1 - \alpha_{j+1}^1 \end{aligned}$$

since $\psi_j(x_1, y_1)$, $\psi_{j+1}(x_2, y_2)$, $\psi_{k_\ell}(x_1, y_1)$, and $\psi_{k_\ell}(x_2, y_2)$ all equal 1. It then follows that $\alpha_j^1 = \alpha_{j+1}^1$. That is, $\mathbf{w}$ is constant by angle.

Finally, since $\mathbf{w} \in NS(A^*)$, it must be that the terms sum to zero, because

$$A^*\mathbf{w}(x, y) = \sum_{k=1}^{M} \alpha_1^k = 0. \qquad \Box$$

An immediate corollary is the following.

COROLLARY 1. *The dimension of the nullspace of B is $M - 1$, where M is the number of projection angles. A basis for $NS(B)$ is*

$$\begin{aligned} \mathbf{w}_1 &= [\vec{1}, -\vec{1}, \vec{0}, \ldots, \vec{0}], \\ \mathbf{w}_2 &= [\vec{0}, \vec{1}, -\vec{1}, \vec{0}, \ldots, \vec{0}], \\ &\vdots \\ \mathbf{w}_{M-1} &= [\vec{0}, \ldots, \vec{0}, \vec{1}, -\vec{1}], \end{aligned}$$

where each constant vector $\vec{1} \in \mathbf{R}^{n(j)}$.

4. Gauss–Seidel relaxation on $B\,\mathbf{w} = \mathbf{f}$. Many iterative methods are available for solving equations in B, and, indeed, many have been applied to the general image reconstruction problem [1], [9], [18], [19], [20], [24]. Here we consider Gauss–Seidel iteration, one sweep of which may be stated as: *For $j = 1 : N$, modify the jth component of the vector* $\mathbf{w}$ *such that the jth component of the resulting residual vanishes.* The jth correction is given by

$$\mathbf{w} \leftarrow \mathbf{w} + \frac{1}{b_{jj}} \langle \mathbf{e}_j, \mathbf{f} - B\mathbf{w} \rangle \mathbf{e}_j.$$

A more common formulation arises from splitting B in terms of its diagonal, upper triangular, and lower triangular parts, giving $B = D - L - U$. Then the $(n + 1)$st sweep may be written as

$$\mathbf{w}^{(n+1)} = P_G \mathbf{w}^{(n)} + \mathbf{g},$$

where $\mathbf{g} = (D - L)^{-1}\mathbf{f}$ is a fixed vector and $P_G = (D - L)^{-1}U$ is known as the Gauss–Seidel *iteration matrix.* Letting $\mathbf{w}^*$ be any vector that solves $B\mathbf{w} = \mathbf{f}$, we may write an *error vector* defined as $\mathbf{z}^{(n)} = \mathbf{w}^{(n)} - \mathbf{w}^*$. It is easy to see that $\mathbf{z}^{(n+1)} = P_G\mathbf{z}^{(n)}$ and, hence, $\mathbf{z}^{(n+1)} = (P_G)^n\mathbf{z}^{(0)}$. Convergence of the iteration to $\mathbf{w}^*$ is guaranteed if the spectral radius $\rho(P_G)$ is less than 1.

The matrix B, however, is rank deficient, so that if any solutions exist, then infinitely many solutions exist, and the iteration does not converge under all initial guesses. However, measured in the *energy seminorm*

$$|||\mathbf{x}||| = \langle B\mathbf{x}, \mathbf{x}\rangle^{1/2},$$

Gauss–Seidel cannot diverge.

THEOREM 3. *The energy seminorm of the error does not increase under Gauss–Seidel iteration on* $B\mathbf{w} = \mathbf{f}$. *That is,* $|||\mathbf{z}^{(n+1)}||| \leq |||\mathbf{z}^{(n)}|||$.

Proof. A direct proof follows from the easily derived relation

$$|||\mathbf{z}^{(n+1)}||| = |||\mathbf{z}^{(n)}||| - \frac{\langle \mathbf{e}_j, B\mathbf{z}^{(n)}\rangle^2}{b_{jj}}. \qquad \square \tag{6}$$

Gauss–Seidel applied to $B\mathbf{w} = \mathbf{f}$ cannot diverge in the energy sense, but to understand when it actually converges we first examine the related Kaczmarz iteration, applied to $Au = \mathbf{f}$.

Proof of the following may be found in [13], [18], and [25].

THEOREM 4. *Let* $L : H_1 \to H_2$ *be a continuous linear operator, where* H_1 *is a Hilbert space and* H_2 *is an* N*-dimensional Hilbert space with orthonormal basis* $\{v_1, v_2, \ldots, v_N\}$. *Let* $g \in H_2$ *be given, and suppose that* $Lu = g$ *has a solution. Suppose* $u^{(0)} \in$ *range*(L^*), *and define the sequence* $u^{(k)}$ *generated by the Kaczmarz iteration by*

```
Set $u \leftarrow u^{(k)}$.
For $j = 1 : N$,
    Determine $s$ such that $\langle v_j, L(u + sL^*v_j) - g\rangle = 0$.
    Set $u \leftarrow u + sL^*v_j$.
Set $u^{(k+1)} \leftarrow u$.
```

Then $u^{(k)}$ *converges, as* $k \to \infty$, *to the minimum norm solution of* $Lu = g$.

Kaczmarz iteration applied to $Au = \mathbf{f}$ uses $H_1 = L_2(\Omega)$, $H_2 = \mathbf{R}^N$ with orthonormal basis $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_N\}$, and

$$s = \frac{\langle \mathbf{e}_j, \mathbf{f} - Au\rangle}{\langle \mathbf{e}_j, AA^*\mathbf{e}_j\rangle}. \tag{7}$$

Proof of the following may be found in [20] or [25].

THEOREM 5. *Let* $\mathbf{w}^{(0)}$ *be any vector in* $\mathbf{R}^N$, *and let* $\mathbf{w}^{(k,j)}$, *for* $j = 1 : N$ *and* $k = 1, 2, \ldots,$ *be the vector resulting from the* j*th step of the* k*th sweep of Gauss–Seidel iteration on* $B\mathbf{w} = \mathbf{f}$, *using* $\mathbf{w}^{(0)}$ *as the initial guess. Then the image* $A^*\mathbf{w}^{(k,j)}$ *is just the image* $u^{(k,j)}$ *resulting from the* j*th step of the* k*th sweep of the Kaczmarz iteration applied to* $Au = \mathbf{f}$ *with initial guess* $u^{(0)} = A^*\mathbf{w}^{(0)}$.

An immediate consequence of this theorem is the following.

COROLLARY 2. *Let* $\mathbf{w}^{(0)}$ *be any vector in* $\mathbf{R}^N$, *and let* $\{\mathbf{w}^{(k)}\}$ *be the sequence of vectors produced by Gauss–Seidel iteration on* $B\mathbf{w} = \mathbf{f}$. *If* $Au = \mathbf{f}$ *has a solution, then the sequence* $A^*\mathbf{w}^{(k)}$ *converges, as* $k \to \infty$, *to the minimum norm solution of* $Au = \mathbf{f}$.

4.1. Numerical performance. We use the positron emission problem, as in PET and SPECT, for our model problem in developing the iterative methods presented here. Such

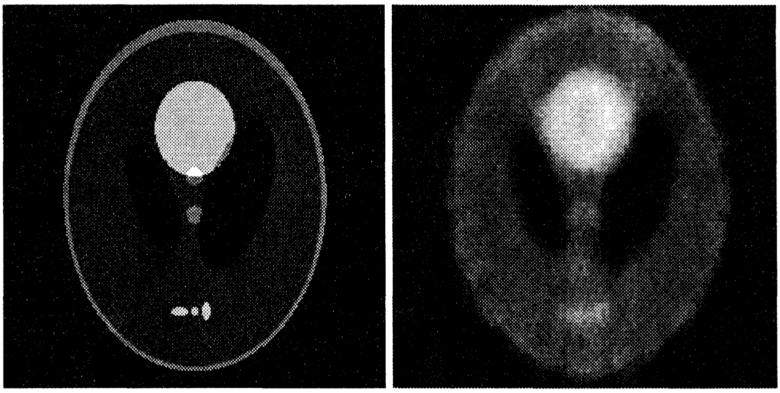


FIG. 3. *An "exact" image is shown on the left, and a reconstructed image is shown on the right. The reconstruction geometry uses data collected in* 64 *bins of uniform width along each of* 20 *angles. The* 20 *angles are equispaced at angular intervals of* $\Delta\phi = \pi/20$ *in the interval* $[0, \pi)$. *Twenty-five sweeps of Gauss–Seidel iteration were used to reconstruct the image.*

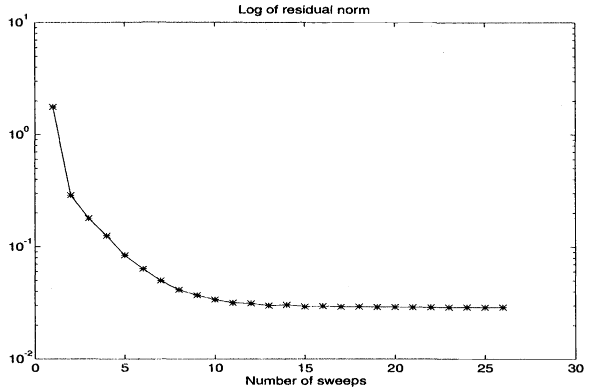


FIG. 4. *Performance of the Gauss–Seidel iteration on* $B\mathbf{w} = \mathbf{f}$ *is displayed by plotting the logarithm of* $\|\mathbf{f} - B\mathbf{w}^{(n)}\|_2$ *as a function of n, the number of iteration sweeps. Twenty-five sweeps of Gauss–Seidel iteration were used to reconstruct the image.*

applications are characterized by relatively small values of N and M, so that we are dealing with fairly small computational problems. Typically, the number of bins per angle, N, is less than 100, as is the number of angles, M. Accordingly, our numerical experiments use $N = 16, 32, 64$, and $M = 10, 20, 64$. Here we report on one such test, which is very representative of the performance characteristics we have observed.

Figure 3 displays an "exact" image, the Shepp–Logan phantom [23], from which a set of values for the right-hand side vector $\mathbf{f}$ is constructed. In generating the vector $\mathbf{f}$ we used $N = 64$ and $M = 20$. Twenty-five sweeps of Gauss–Seidel on $B\mathbf{w} = \mathbf{f}$ produce the reconstructed image shown in 3.

Qualitatively, one can argue that the procedure produces a good reconstruction, in that most of the identifiable features of the original image are present. Since, in general, the exact image is unknown, we use the residual $\mathbf{f} - B\mathbf{w}$ as a numerical indication of how well the method solves the problem. Figure 4 displays the logarithm of $\|\mathbf{f} - B\mathbf{w}^{(n)}\|_2$ as a function of n, the number of iteration sweeps. Noteworthy is the fact that the first few sweeps result in significant reduction in the norm of the residual, but that the improvement per

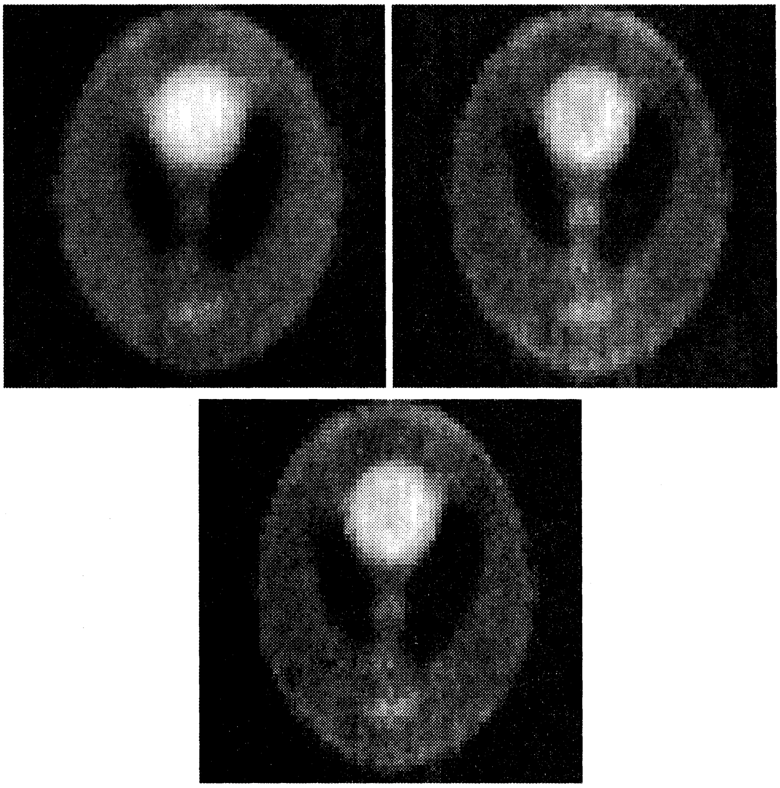


FIG. 5. *Reconstructions of the image from Fig.* 3 *using* 1, 2, *and* 4 *Gauss–Seidel sweeps are displayed clockwise from the upper left. It is difficult to distinguish these reconstructions, and even more difficult to determine which is "best."*

sweep declines until (after approximately 10 sweeps) the residual norm remains essentially unchanged.

This behavior, of rapid improvement in the residual norm over the course of several sweeps followed by stagnation of the residual norm, is characteristic of many iteration methods. In the field of partial differential equations, this numerical "stalling" often occurs because relaxation eliminates the oscillatory components of the error rapidly, but is ineffectual on the remaining smooth components of the error. The stalling phenomenon is often eliminated through the use of multigrid algorithms. Shortly we will develop a multigrid method for the problem $B\mathbf{w} = \mathbf{f}$, in an attempt to address the numerical stalling. Before doing so, however, we wish to make two observations.

First, the stalling phenomenon is often unrelated to the quality of the reconstructed image when the quality is measured by the subjective standard of "looking good." While this measure is hard to quantify, and therefore not so useful to the mathematician or engineer, it is the ultimate measure applied by the end user, for example, the radiologist tasked with treating a patient. It is important to note that the reconstructed images frequently *look* good after only one or two iteration sweeps, while the numerical stalling is not apparent until much later. Figure 5 shows reconstuctions of the "exact" image of Fig. 3 as they appear after 1, 2, and 4 sweeps. While subtle differences are apparent in the reconstructions, all are "good." Indeed, it is difficult to differentiate the reconstructions after 4 sweeps and 25 sweeps (Fig. 3). For this reason, the residual norm may not be the appropriate indicator of reconstruction quality.

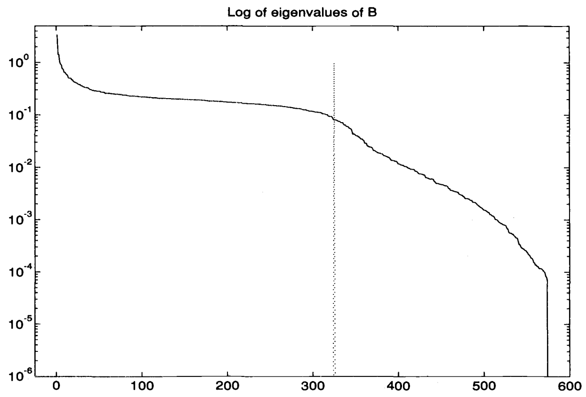


FIG. 6. *The logarithms of the eigenvalues of a typical matrix B are shown. $M = 20$ angles are used, so that the null space of B has dimension* 15. *The eigenvalues between index* 1 *and index* 325 *are the "good" modes, while those between* 326 *and* 573 *are the* near null space eigenvalues.

4.2. Mode analysis. The second observation we make is that it is possible to examine the performance of the Gauss–Seidel iteration on individual components of the error. For numerical partial differential equations this is often done by way of Fourier analysis [4]. However, Fourier analysis is not particularly useful in this setting, because the Fourier modes are not eigenfunctions of the continuum operator, nor are discrete Fourier modes eigenvectors of either the matrix B or the iteration matrix P_G.

The approach we take is somewhat empirical in nature: we examine the eigenvalues and corresponding eigenvectors of the matrix B. Since B is singular with rank $N - M + 1$, we know that zero is an eigenvalue of multiplicity $M - 1$. We are not concerned with eigenvectors corresponding to the zero eigenvalues, as they have no impact on the norm of the residual or on the reconstruction itself (their backprojections vanish).

Slow convergence of the iteration implies that the correction given by the iteration is insufficient. Since the size of the correction to the jth unknown is determined by the jth entry in the residual, then slow convergence implies that the residual is "small" compared to the error. Indeed, this has been shown to be the case for many familiar iteration schemes [3]. Since B applied to the error gives the residual, troublesome components are thus errors consisting essentially of the eigenvectors associated with the small nonzero eigenvalues of B, the near null space components.

Figure 6 displays a plot of the eigenvalues of a representative B matrix, which is typical of the spectra of all matrices we have examined. The geometry for this case has $M = 20$ angles and $N = \sum_{j=1}^{20} n(j) = 592$ total detectors. The set of eigenvalues is divided into three groups: the zero eigenvalues, the large nonzero eigenvalues, and the group of small eigenvalues λ_k whose amplitudes decay rapidly with increasing index k. The last 19 eigenvalues, λ_{574} through λ_{592}, are zero, and the associated eigenvectors form $NS(B)$. The vertical dashed line in the figure marks the division between the "good" eigenvalues (λ_1 through λ_{325}) and the eigenvalues with rapidly decaying magnitude (λ_{326} through λ_{573}), whose associated eigenvectors form the near null space. These near null space eigenvectors are the "slow" modes that stall performance.

We show this empirically in the following way. Representative modes are selected from the "good" and near null space segments of the spectrum. Each such mode is used as the right-hand side of $B\mathbf{w} = \mathbf{f}$. Gauss–Seidel relaxation is then applied, with an initial guess of 0, to solve the equation. Two important observations are obtained in this way. First, by computing the norm of the residual at the end of each sweep, we may determine the convergence factor for

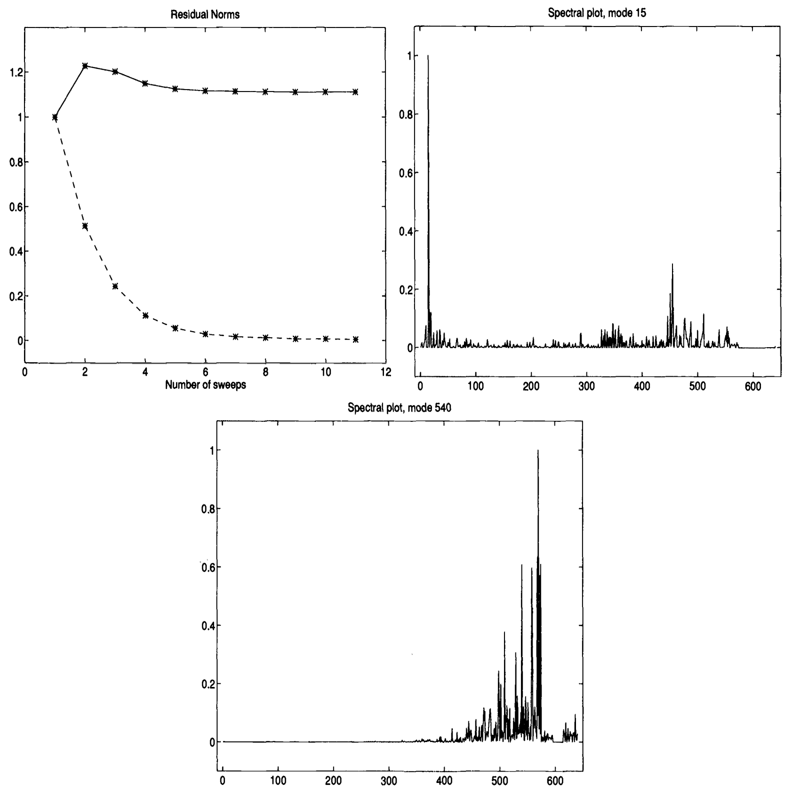


FIG. 7. *The residual norms are shown* (*top left*) *for Gauss–Seidel applied to* $B\mathbf{w} = \mathbf{f}$ *using eigenvectors as the right-hand side. Observe that for mode* 15 (*dashed line*) *the iteration converges well, while for mode* 540 *the iteration stalls immediately. The spectral densities of the vectors resulting from one relaxation sweep is shown for mode* 15 (*top right*) *and mode* 540 (*bottom*).

each mode. Second, after one sweep of Gauss–Seidel, we compute the projection of the current approximation in the directions of all the eigenvectors of B, and plot the resulting magnitudes against the eigenvalue index. This results in a "power spectral density" of the latest iterate. Since the initial error in such an experiment is in the direction of a single eigenvector (the right-hand side), such a spectral density plot tells us to what extent the iteration mixes modes, and if the iteration excites some of the modes, to which part of the spectrum they belong.

Figure 7 shows a typical set of results of these experiments. On the top left are residual norms, as a function of iteration sweeps, for two eigenvectors. The dashed line shows the residual norms for the eigenvector corresponding to λ_{15}, a typical "good" mode, while the solid line gives the residual norms for the eigenvector corresponding to λ_{540}, a typical near null space mode. In the top right is shown the power spectral density plot after one sweep with the 15-mode as the right-hand side, while on the bottom is the corresponding spectral density plot for the 540-mode.

The results shown in Figure 7 support our assertion regarding the modes. That is, the iteration attenuates "good" error modes rapidly, while the iteration stalls on near null space error modes. In addition, we see that there is mode mixing in the spectral density plots for both cases, but that it is much more pronounced in the case of mode 540. In both cases, though, the excitation of extraneous modes occurs predominantly in the near null space band.

Finally, the images resulting from backprojecting modes 15 and 540 are shown in Fig. 8, and again they are typical cases. The backprojected "good" modes generally appear as smooth,

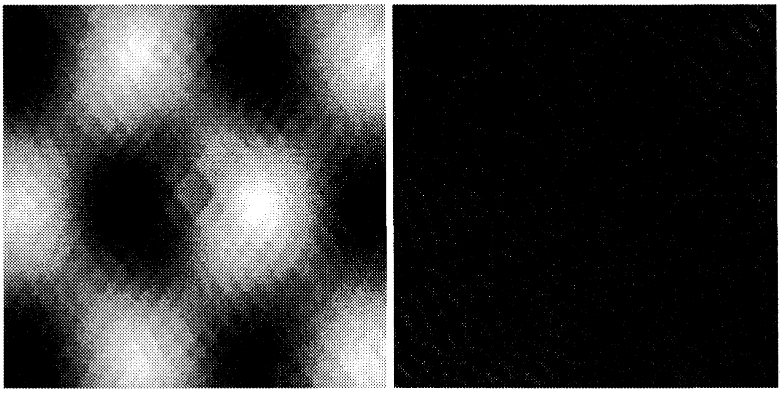


FIG. 8. *The images corresponding to backprojecting mode* 15 (*left*) *and mode* 540 (*right*) *are shown. The "good" mode is a gently undulating surface, while the near null space mode is nearly black.*

highly geometric structures in image space, often as gently undulating surfaces. The backprojected near null space modes, as the name implies, are almost invisible. They often show distinct geometric characteristics, such as narrow subparallel striping, or isolated spikes in nearly flat images.

At this stage we have a good idea what the Gauss–Seidel method achieves. Further, we have a fairly complete picture of where and why it stalls. We next develop multilevel methods for solving $B\mathbf{w} = \mathbf{f}$. We do this for three reasons. First, we believe that if the nature of the near null space modes can be accurately determined, it may be possible to design a "coarse grid correction" to treat the bad modes of the error. That is, we may be able to find a grid on which the bad modes can be annihilated efficiently. Hence, we may hope to achieve multigrid acceleration on this problem. Since the bad modes are not characterized by physical smoothness, like they are in model multigrid problems, standard coarsening is not likely to be very effective. As with other applications that do not possess standard smoothness properties, we must be careful to use what we know about these bad components to devise a coarsening process that closely matches them. Our first attempt is based on the presumption that, at certain scales (e.g., when the number of angles is small compared to the resolution within projections), they must be smooth within projections. We base our multigrid scheme on this idea. A second and equally important reason for using multigrid is our anticipation of the spotlight tomography problem, introduced in the final section of this paper. Finally, the correct isolation of these modes that occurs naturally and efficiently in multilevel processing may pave the way for treating them by an individualized regularization process that is better tailored to the computational objectives.

5. Multilevel image reconstruction.

5.1. Multilevel projection methods (PML). Designing a multigrid method for a new problem is a difficult task, especially when the application is far removed from the classical multigrid setting of elliptic PDEs. Multigrid has been extended to a wide variety of such problems, with varying degrees of success [6], [17], [22]; multigrid design in such instances is generally a lengthy and difficult process.

The multilevel projection methodology (PML) was developed to provide a simpler, systematic approach to multilevel algorithm design [15]. A basic tenet of PML design is that only the appropriate subspaces in which the problem is to be set need to be determined. The problem is discretized by orthogonal projections, and the projection operators in turn lead to the correct choices for intergrid transfer operators, relaxation techniques, and coarsening schemes.

To briefly describe the fundamentals of PML, let H_1 and H_2 be Hilbert spaces and $L : H_1 \to H_2$ be a linear operator. The continuum problem is to find $u \in H_1$ such that $Lu - f = 0$. *Discretization by projections* is accomplished as follows.

Let S^h be a finite-dimensional subspace of H_1, and let $P^{S^h} : H_1 \to S^h$ be an orthogonal projection of H_1 onto S^h, where the superscript h refers to a discretization parameter. We also require a finite-dimensional subspace $T^h \subset H_2$, and an orthogonal projection $P^{T^h} : H_2 \to T^h$, as well as mappings $P_{S^h} : S^h \to H_1$ and $P_{T^h} : T^h \to H_2$.

The projection operators are used to generate a discrete operator $L^h : S^h \to T^h$ by projecting the action of the continuum operator L onto the subspaces, that is, the discretized problem becomes $P^{T^h} L(P^{S^h} u) = 0$, for $u \in H_1$. This allows us to define the discrete operator for the problem by

$$L^h u^h - f^h = 0, \quad \text{for } u^h \in S^h, \qquad \text{where} \qquad L^h \equiv P^{T^h} L P^{S^h} \text{ and } f^h = P^{T^h} f.$$

We pause here to show that the strip pixel discretization developed earlier is in fact a discretization by projection.

THEOREM 6. *For each $j = 1 : M$, let Ω be exactly partitioned into $n(j)$ parallel nonoverlapping strips and let $N = \sum_{j=1}^{M} n(j)$. Number the strips from 1:N and let $\psi_j(x, y)$ be the characteristic function of the jth strip. Let S^h be the subspace of the Hilbert space $H_1 = L_2(\Omega)$ spanned by the set*

$$\{\psi_j\}_{j=1}^{N}.$$

Then the matrix equation

$$B\mathbf{w} = \mathbf{f}$$

is a discretization by projections of the problem $Au = \mathbf{f}$, where A is the strip-averaged Radon transform (1) *and B is the $N \times N$ matrix with entries $b_{jk} = \langle \psi_j, \psi_k \rangle$.*

Proof. We define the various subspaces of the discretization as follows. H_1 and S^h are defined in the statement of the theorem. Note that (3) implies that $S^h = \text{range}\{A^*\}$. We take $H_2 = \mathbf{R}^N$ and define the subspace T^h to be H_2. Since $H_2 = T^h = \mathbf{R}^N$, we may take $P^{T^h} = I_N$, the $N \times N$ identity matrix. The discrete equation will then be $AP^{S^h} u = \mathbf{f}$, where $P^{S^h} u$ is the orthogonal projection of $u(x, y)$ onto S^h, so

$$P^{S^h} u = \sum_{k=1}^{N} w_k \psi_k(x, y) = A^{h*}\mathbf{w}$$

for some $\mathbf{w} \in \mathbf{R}^N$. Since P^{S^h} is an orthogonal projection, we must have $(u - P^{S^h} u) \perp \psi_j$ for every $\psi_j \in S^h$. Hence, for $j = 1 : N$,

$$\begin{aligned} 0 &= \left\langle u - P^{S^h} u, \psi_j \right\rangle \\ &= \left\langle u - \sum_{k=1}^{N} w_k \psi_k, \psi_j \right\rangle \\ &= \left\langle u, \psi_j \right\rangle - \sum_{k=1}^{N} w_k \left\langle \psi_k, \psi_j \right\rangle. \end{aligned}$$

Thus, if $P^{S^h} u = A^{h*}\mathbf{w}$ is an orthogonal projection of $u(x, y)$ into S^h, then the vector $\mathbf{w}$ must satisfy

$$[\langle u, \psi_1 \rangle \ \langle u, \psi_2 \rangle, \dots, \langle u, \psi_N \rangle]^T = B\mathbf{w}. \tag{8}$$

But the left-hand side of (8) is just Au, so $\mathbf{w}$ must solve $Au = B\mathbf{w}$, and the projection-discretized form of $Au = \mathbf{f}$ is just $B\mathbf{w} = \mathbf{f}$. □

Henceforth, to keep track of the level we are examining, we use the notation $B^h\mathbf{w}^h = \mathbf{f}^h$, where B^h is the matrix defined by (4). We also adopt superscripts for use with the characteristic functions of the strips, e.g., ψ^h_k.

An important observation to be made here is that it is not necessary to know the projection operators explicitly if the *condition* of orthogonal projection adequately defines the discrete operator L^h.

Now we can examine how the PML method makes use of discretization by projections to build a two-level solver. Let $P^{S^{2h}}$ and $P^{T^{2h}}$ be projection operators mapping the continuum spaces H_1 and H_2 into "coarse grid" subspaces $S^{2h} \subset S^h \subset H_1$ and $T^{2h} \subset T^h \subset H_2$. The coarse grid operator is given by $L^{2h} = P^{T^{2h}} L P^{S^{2h}}$.

The two main components of any multigrid problem are relaxation and coarse grid correction. The PML approach defines relaxation by decomposing the spaces S^h and T^h into sums (which need not be direct sums) of m subspaces

$$S^h = \sum_{\ell=1}^{m} S^h_\ell \qquad \text{and} \qquad T^h = \sum_{\ell=1}^{m} T^h_\ell .$$

Any element of S^h can be written as a linear combination (not necessarily unique!) of the elements of S^h_ℓ:

$$u^h = \sum_{\ell=1}^{m} \alpha_l u^h_{(\ell)}, \quad \text{where } u^h_{(\ell)} \in S^h_\ell .$$

The ℓth relaxation step is defined by adding to the current approximation u^h an element of the subspace S^h_ℓ such that the projection of the residual into T^h_ℓ vanishes. A relaxation sweep is made by performing the relaxation step for all m subspaces. Hence, *relaxation by projections* is defined by

```
Relaxation:    $u^h \leftarrow G^h(u^h)$
    For $\ell = 1, 2, \ldots, m$
        1. Determine $u^h_{(\ell)} \in S^h_\ell$ such that $P^{T^h_\ell} L^h(u^h + u^h_{(\ell)}) - f^h = 0$.
        2. Set $u^h \leftarrow u^h + u^h_{(\ell)}$.
```

In standard multigrid, coarse grid correction is performed by restricting the residual equation to the coarse grid, solving for the error, interpolating the error to the fine grid, and adding it to the fine grid approximation. This basic process is also what PML does, though in an abstract way that is guided by the discretization. Given the coarse level subspaces $S^{2h} \subset S^h \subset H_1$ and $T^{2h} \subset T^h \subset H_2$, together with the associated projection operators, the aim is to determine an element of the coarse space S^{2h} that, when added to the current approximation u^h, satisfies the projection of the residual equation onto T^{2h}. Thus, *coarse grid correction by projections* is written as

```
Coarse grid correction: $u^h \leftarrow C^h(u^h)$
    1. Determine $u^{2h} \in S^{2h}$ such that

        $P^{T^{2h}} \left( L(P^{S^h} u^h + P^{S^{2h}} u^{2h}) - f^h \right) = 0.$

    2. Set $u^h \leftarrow u^h + u^{2h}$.
```

Relaxation and coarse grid correction together form a two-level PML method that is given by

```
Two-level PML method:    $u^h \leftarrow PML^h(u^h)$
    • $u^h \leftarrow G^h(u^h)$.
    • $u^h \leftarrow C^h(u^h)$.
```

The two-level PML algorithm can be converted into a multilevel scheme in just the same way that standard multigrid schemes are developed from two-grid schemes: the exact solver in the coarse grid correction is replaced by a recursive call $u^{2h} \leftarrow PML^{2h}(u^{2h})$, leading to a PML V-cycle, for example.

6. PML image reconstruction.

6.1. Discretization and intergrid transfers. Applying PML to equations in A is somewhat subtle [15], primarily because of the need to treat both projection (Radon transform) and image spaces, but with the optimal pixel discretization, applying PML to equations in $B = AA^*$ may be more direct. We have already shown that $B^h \mathbf{w}^h = \mathbf{f}^h$ is a discretization by projections of the problem $Au = \mathbf{f}$ onto S^h and T^h. It is easy to define coarse subspaces S^{2h} and T^{2h} in a manner that leads to a useful multilevel algorithm.

Let S^h be the span of the N strip pixels ψ_j^h, where the h is some parameter that indicates the level of the discretization (e.g., h may be the width of the widest strip pixel). Suppose for simplicity that there is an even number of strip pixels for each of the M views, and that we number the strips from ψ_1^h to ψ_N^h in a way so that two adjacent strips on any view are always numbered consecutively. Then a useful subspace S^{2h} can be constructed according to

$$(9) \qquad S^{2h} = \operatorname{span}\left\{\psi_k^{2h}\right\}_{k=1}^{N/2} \qquad \text{where} \qquad \psi_k^{2h} = \psi_{2k-1}^h + \psi_{2k}^h .$$

Thus, each strip pixel in the coarse subspace is the union of two adjacent fine space strip pixels. This may be viewed in physical terms as widening the aperture of the detectors, or bins.

With these coarse space strip pixels, we find that $A^{2h^*}\mathbf{w}^{2h} = \sum_{j=1}^{N/2} w_j^{2h} \psi_j^{2h}$, from which we easily obtain $A^{2h} : S^{2h} \to \mathbf{R}^{N/2}$ by $\left(A^{2h}u\right)_i = \langle \psi_i^{2h}, u \rangle$. This in turn leads us to the projection discretized coarse level problem $B^{2h}\mathbf{w}^{2h} = \mathbf{f}^{2h}$, where $B^{2h} = A^{2h}A^{2h^*}$ is an $N/2 \times N/2$ matrix with entries $b_{ij}^{2h} = \langle \psi_i^{2h}, \psi_j^{2h} \rangle$.

The multilevel scheme requires interlevel transfer operators to map grid functions between the coarse and fine levels, and a basic tenet of PML is that these operators are defined implicitly. That is, $P_{S^h}^{S^{2h}} : S^h \to S^{2h}$ and $P_{S^{2h}}^{S^h} : S^{2h} \to S^h$ are defined by

$$P_{S^{2h}} = P_{S^h} P_{S^{2h}}^{S^h}, \qquad \text{and} \qquad P^{S^{2h}} = P_{S^h}^{S^{2h}} P^{S^h}.$$

Analogous transfers can be defined for the subspaces $T^{2h} \subset T^h \subset H_2$.

To determine the intergrid transfer operators $P_{S^h}^{S^{2h}}$ and $P_{S^{2h}}^{S^h}$, we begin with a simple observation based on the definition of the coarse space strips ψ_k^{2h}.

LEMMA 1. *The operators $A^h : S^h \to T^h$ and $A^{2h} : S^{2h} \to T^{2h}$ are related by*

$$A^{2h} = P_{S^h}^{S^{2h}} A^h$$

where $P_{S^h}^{S^{2h}}$ is an $N/2 \times N$ matrix given by

$$P_{S^h}^{S^{2h}} = \begin{pmatrix} 1 & 1 & & & & & \\ & & 1 & 1 & & & \\ & & & & \ddots & \ddots & \\ & & & & & & 1 & 1 \end{pmatrix}.$$

Furthermore, the adjoint operators $A^{h^*} : T^h \to S^h$ and $A^{2h^*} : T^{2h} \to S^{2h}$ are related by

$$A^{2h^*} = A^{h^*} \left(P_{S^h}^{S^{2h}}\right)^T .$$

Proof. Let the coarse grid strip pixel ψ_k^{2h} be the union of adjacent fine grid strip pixels given by $\psi_k^{2h} = \psi_{2k-1}^h + \psi_{2k}^h$, for $k = 1 : N/2$. Then

$$\langle \psi_k^{2h}, u\rangle = \langle \psi_{2k-1}^h + \psi_{2k}^h, u\rangle = \langle \psi_{2k-1}^h, u\rangle + \langle \psi_{2k}^h, u\rangle = (1 \quad 1)\begin{pmatrix} \langle \psi_{2k-1}^h, u\rangle \\ \langle \psi_{2k}^h, u\rangle \end{pmatrix}.$$

The first assertion of the lemma follows by partitioning the vector $A^h u$ into blocks consisting of pairs of adjacent entries and forming the matrix $P_{S^h}^{S^{2h}}$ by placing, for $k = 1 : N/2$, the block $(1 \quad 1)$ in the $(2k-1)$st and $(2k)$th positions of the kth row of an $N/2 \times N$ zero matrix. Matrix vector multiplication then yields $A^h u$.

The second assertion is established by

$$\begin{aligned} A^{2h*}\mathbf{w}^{2h} &= \begin{pmatrix} \psi_1^{2h} & \psi_2^{2h} & \cdots & \psi_{N/2}^{2h} \end{pmatrix} \mathbf{w}^{2h} \\ &= \begin{pmatrix} \psi_1^h + \psi_2^h & \psi_3^h + \psi_4^h & \cdots & \psi_{N/2-1}^h + \psi_{N/2}^h \end{pmatrix} \mathbf{w}^{2h} \\ &= \left[A^{h*} \left(P_{S^h}^{S^{2h}} \right)^T \right] \mathbf{w}^{2h}. \qquad \square \end{aligned}$$

The second part of the lemma verifies that the operator $P_{S^h}^{S^{2h}}$ gives a consistent definition to the adjoint of the coarse space operator A^{2h}, showing that $A^{2h*} = (P_{S^h}^{S^{2h}} A^h)^* = A^{h*}(P_{S^h}^{S^{2h}})^T$. Thus we may define $P_{S^{2h}}^{S^h} = (P_{S^h}^{S^{2h}})^T$. Combining this with the observation that $B^{2h} = A^{2h}A^{2h*} = P_{S^h}^{S^{2h}} A^h A^{h*} P_{S^{2h}}^{S^h} = P_{S^h}^{S^{2h}} B^h P_{S^{2h}}^{S^h}$, we find that the standard variational conditions of multigrid are satisfied by this discretization [5].

6.2. Relaxation. Following the principles of PML, we select sets of m subspaces S_ℓ^h and T_ℓ^h whose unions equal S^h and T^h, respectively. The most obvious choice is to select $m = N$ and $S_\ell^h = \operatorname{span}\{\psi_\ell^h\}$, that is, each subspace is the span of an individual strip pixel.

LEMMA 2. *Let $S_\ell^h = \operatorname{span}\{\psi_\ell^h\}$ and $T_\ell^h = \operatorname{span}\{\mathbf{e}_\ell^h\}$ for $\ell = 1 : N$, where $\mathbf{e}_\ell^h$ is the ℓ^{th} standard basis vector in T^h. Then the PML relaxation on $A^h u^h = \mathbf{f}^h$ is implemented by performing point Gauss–Seidel iteration on the matrix equation $B^h\mathbf{w}^h = \mathbf{f}^h$.*

Proof. The ℓth step of PML relaxation consists of finding that value of α satisfying

$$P_\ell^{T^h}\left(AP^{S^h}(u^h + \alpha\psi_\ell^h) - \mathbf{f}^h\right) = 0,$$

where u^h is the current approximation to the solution, and then modifying the approximation by $u^h \leftarrow u^h + \alpha\psi_\ell^h$. Now $\psi_\ell^h = A^{h*}\mathbf{e}_\ell^h$, and since $P^{S^h}u^h \in S^h$, then we know that $P^{S^h}u^h = A^{h*}\mathbf{w}^h$ for some $\mathbf{w}^h \in T^h$. Hence we seek α such that

$$\begin{aligned} 0 &= P_\ell^{T^h}\left(A^h(A^{h*}\mathbf{w}^h + \alpha A^{h*}\mathbf{e}_\ell^h) - \mathbf{f}^h\right) \\ &= P_\ell^{T^h} B^h(\mathbf{w}^h + \alpha\mathbf{e}_\ell^h) - f_\ell^h. \end{aligned}$$

Noting that $P_\ell^{T^h}$ is accomplished by forming the inner product with $\mathbf{e}_\ell^h$, then α must satisfy

$$(\mathbf{e}_l^h)^T(B^h\mathbf{w}^h + \alpha B^h\mathbf{e}_\ell^h) = f_\ell,$$

whose solution is given by

$$\alpha = \frac{1}{b_{\ell\ell}}(f_\ell^h - \mathbf{b}_\ell^T\mathbf{w}^h),$$

where $\mathbf{b}_\ell^T$ is the ℓth row of B^h. Hence, the ℓth step of the PML relaxation is

$$\mathbf{w}_\ell^h \leftarrow \mathbf{w}_\ell^h + \frac{1}{b_{\ell\ell}}(f_\ell^h - \mathbf{b}_\ell^T \mathbf{w}^h),$$

which is precisely the correction of the ℓth step of Gauss–Seidel applied to $B^h \mathbf{w}^h = \mathbf{f}^h$. □

6.3. Coarse grid correction. Like relaxation, coarse grid correction in the PML approach is defined by the selection of the subspaces and the implicit intergrid transfer operators. For the problem $Au = \mathbf{f}$, it is performed by finding the element $u^{2h} \in S^{2h}$ that satisfies

$$P^{T^{2h}}\left(A(P^{S^h}u^h + P^{S^{2h}}u^{2h}) - \mathbf{f}^h\right) = 0, \tag{10}$$

where u^h is the current approximation in the fine space S^h. The correction is then given by $u^h \leftarrow u^h + u^{2h}$. Note that

$$P^{T^{2h}} A P^{S^h} u^h = P_{S^h}^{S^{2h}} P^{T^h} A P^{S^h} u^h = P_{S^h}^{S^{2h}} B^h \mathbf{w}^h$$

where $A^{h*}\mathbf{w}^h$ represents $P^{S^h}u^h$. We also know that since $u^{2h} \in S^{2h}$, there is a vector $\mathbf{y}^{2h} \in T^{2h}$ such that $u^{2h} = A^{2h*}\mathbf{y}^{2h}$. Hence, $P^{T^{2h}} A P^{S^{2h}} u^{2h} = B^{2h}\mathbf{y}^{2h}$. Noting also that $P^{T^{2h}}\mathbf{f} = P_{S^h}^{S^{2h}}\mathbf{f}$, then (10) becomes

$$P_{S^h}^{S^{2h}} B^h \mathbf{w}^h + B^{2h}\mathbf{y}^{2h} - P_{S^h}^{S^{2h}} \mathbf{f}^h = 0.$$

The correction step is thus $\mathbf{w}^h \leftarrow \mathbf{w}^h + P_{S^{2h}}^{S^h}\mathbf{y}^{2h}$. Hence, with the operators we have constructed, the PML coarsening step for this problem is formally the same as conventional multigrid:

1. Set $\mathbf{f}^{2h} = P_{S^h}^{S^{2h}}(\mathbf{f}^h - B^h\mathbf{w}^h)$.
2. Solve $B^{2h}\mathbf{y}^{2h} = \mathbf{f}^{2h}$.
3. Correct the approximation by $\mathbf{w}^h \leftarrow \mathbf{w}^h + P_{S^{2h}}^{S^h}\mathbf{y}^h$.

Of course, as with any multigrid algorithm, in practice the exact solve on the coarse grid is replaced by a recursion, so that the only time an exact solution is computed is on the coarsest subspace. To form such a recursion in the strip pixel PML setting, we need only continue defining coarser spaces S^{jh}, for $j = 1, 2, \ldots$. This is done by taking the strip pixels that generate the new subspace to be the pairwise union of strip pixels in the current subspace, just as was done to produce S^{2h} from S^h. Once this is done, a PML V-cycle can be defined in the usual way.

```
PML V-cycle:    $\mathbf{w}^h \leftarrow PMLV(\mathbf{w}^h, B^h, \mathbf{f}^h)$
1. Relax $\nu_1$ times on $B^h\mathbf{x}^h = f^h$ with initial guess $\mathbf{w}^h$.
2. If $S^h$ represents the coarsest level, go to 3. Otherwise:
    (a) $\mathbf{f}^{2h} \leftarrow P_{S^h}^{S^{2h}}(f^h - B^h\mathbf{w}^h)$
    (b) $\mathbf{w}^{2h} \leftarrow 0$.
    (c) $\mathbf{w}^{2h} \leftarrow PMLV(\mathbf{w}^{2h}, B^{2h}, \mathbf{f}^{2h})$
    (d) $\mathbf{w}^h \leftarrow \mathbf{w}^h + P_{S^{2h}}^{S^h}\mathbf{w}^{2h}$.
3. Relax $\nu_2$ times on $B^h\mathbf{x}^h = f^h$ with initial guess $\mathbf{w}^h$.
```

6.4. Numerical performance. Figures 9 and 10 display two examples of the image reconstructions obtained with the PMLV algorithm. The pair of images in Fig. 9 were obtained using 20 views with 32 detectors per view, and restricting the image to lie in the unit square. The data were generated by projecting the exact image on the left, while the reconstruction of the image by PMLV is shown on the right. The reconstruction was made using 3 PMLV cycles with 2 relaxation sweeps on the downward leg of the V and one relaxation sweep on

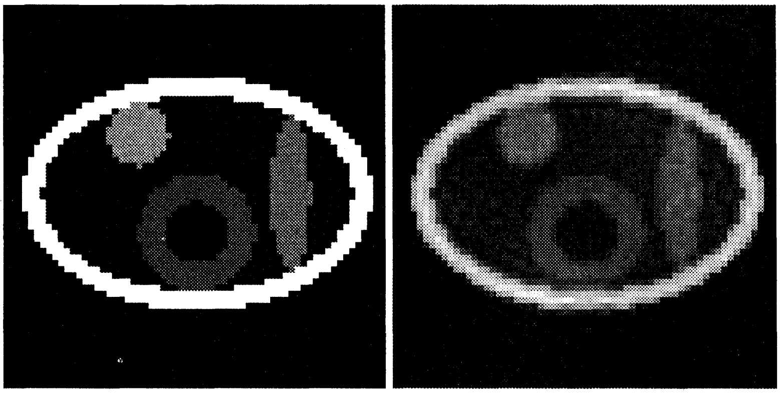


FIG. 9. *The performance of the PMLV method may be observed by comparing the actual (left) and reconstructed (right) images of a "brain phantom." The reconstruction was obtained using* 3 *PMLV cycles with* 2 *relaxation sweeps on the downward leg of the V and one relaxation sweep on the upward leg. The image on the left was used to generate the data used in the reconstruction, with* 20 *angles and* 32 *detectors per angle.*

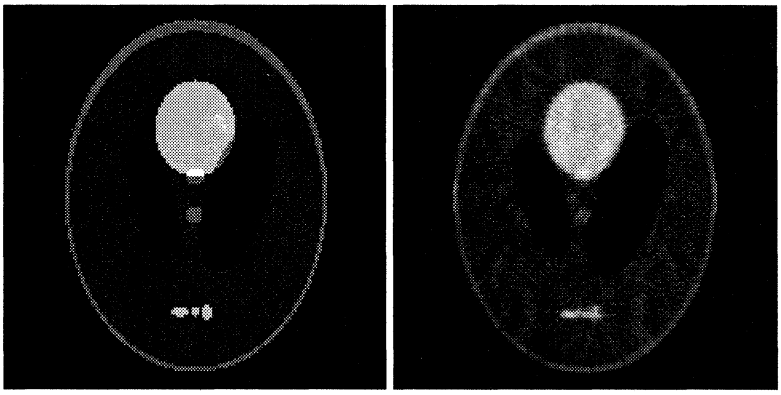


FIG. 10. *The reconstruction on the right was obtained using the multilevel method from data generated from the image on the left, using* 64 *angles and* 64 *detectors per angle. The reconstruction was obtained using* 3 *PMLV cycles with* 2 *relaxation sweeps on the downward leg of the V and one relaxation sweep on the upward leg.*

the upward leg. The Shepp–Logan phantom was used for the reconstruction in Fig. 10, which was obtained from 64 views with 64 detectors per view. Again, the image on the left was used to generate data for the multilevel reconstruction (right), which was made from 3 PMLV (2,1)-cycles.

To compare the performance of PMLV and Gauss–Seidel, let the work entailed by one sweep of Gauss–Seidel on the finest level be one work unit (WU), which is an $O(N^2)$ operation. Computation of the residual requires approximately one WU. The work required for one PMLV-cycle, using $\nu_1 + \nu_2$ sweeps of Gauss–Seidel on each level, is then bounded by

$$(\nu_1 + \nu_2 + 1)\left(1 + \frac{1}{4} + \frac{1}{16} + \cdots\right) \approx \frac{4}{3}(\nu_1 + \nu_2 + 1).$$

PMLV was applied to several reconstruction problems, using several different geometries. The performance of the algorithm in all tests was similar, and may be summarized by examining the results of a typical suite of experiments. In these tests the parameters $\nu_1 = 2$ and $\nu_2 = 1$ give the number of iteration sweeps, respectively, descending and ascending through the

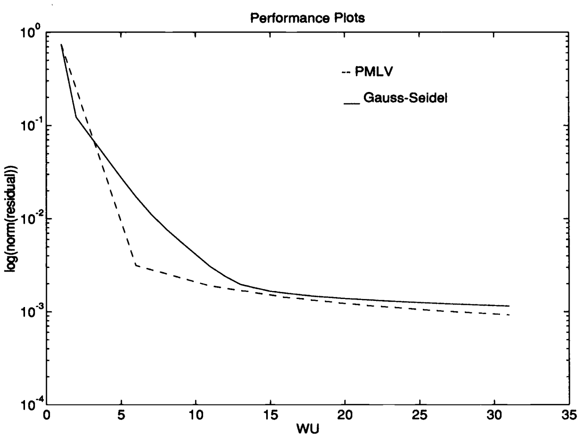


FIG. 11. *Comparison of the performance of Gauss–Seidel and PMLV. The logarithm of the residual norm is plotted as a function of the number of work units required to attain it.*

V-cycle. Hence, one V-cycle requires approximately $\frac{16}{3}$ WUs. The problem was coarsened to the coarsest possible level, giving one strip per view and a problem of size $M \times M$ at the coarsest level. Figure 11 compares the typical performance of Gauss–Seidel to the PMLV algorithm for a problem with 32 detectors over 20 angles.

It is clear from Fig. 11 that, even for this relatively small problem, PMLV initially outperforms Gauss–Seidel. However, continued iteration of Gauss–Seidel eventually achieves similar results at similar costs. We believe this is due largely to the fact that the bad modes do not possess the physical smoothness characteristic of bad modes in elliptic PDE problems, so that coarsening by row-lumping within projections does not entirely succeed at eliminating the bad modes. We think that this may be caused by the problem entering a scale regime where there is close coupling between the projections. This is likely to mean that a special lumping of rows is needed, where the oscillatory but possibly regular pattern of these components across angles is taken into account. Note that the slopes at the right end of the curves indicate that further iteration may favor PMLV. However, it is important to recall that the ultimate goal is quality image reconstruction, and that the residual norm may not be a reliable measure of success. It is also important to recall the ill-posed nature of reconstruction. Consider the problem $B\mathbf{w} = \mathbf{f}$, whose exact solution (in the least squares sense) is

$$\mathbf{w} = B^{\dagger}\mathbf{f},$$

where $B^{\dagger}$ is the pseudoinverse of B. In terms of the singular value decomposition $B = U\Sigma V^T$, the solution $\mathbf{w}$ can be expressed as

$$\mathbf{w} = \left[\sum_{i=1}^{r} \frac{1}{\sigma_i}\mathbf{v}_i\mathbf{u}_i^T\right]\mathbf{f} = \sum_{i=1}^{r} \frac{1}{\sigma_i}\mathbf{v}_i\langle\mathbf{u}_i, \mathbf{f}\rangle,$$

where the σ_i's are the singular values, $\mathbf{u}_i$ and $\mathbf{v}_i$ (the left and right singular vectors of B) are the columns of U and V, and $r = \operatorname{rank}(B)$. If there is *measurement noise* in the data, so that instead of $\mathbf{f}$ the data are $\mathbf{f} + \varepsilon$, then solution components corresponding to small singular values will magnify this noise. These are the components in the near null space that are slow to be recovered. Thus, continued iteration after the procedure stalls in an attempt to recover these slow components has the potential to corrupt the solution with magnified noise [11], [19].

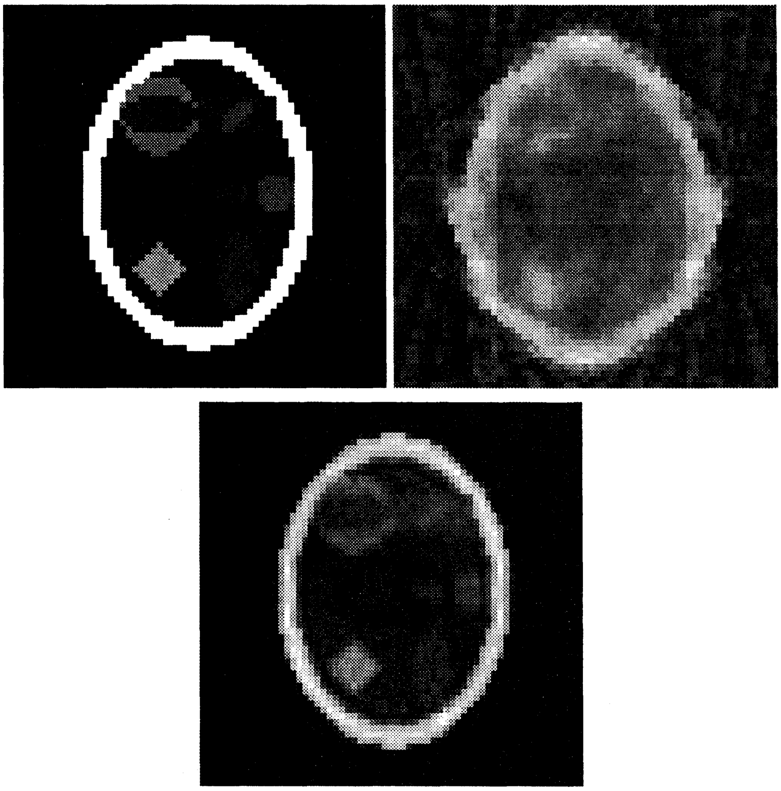


FIG. 12. *Exact image (top left), minimum norm solution (top right), and PMLV solution (bottom).*

Experiments have shown, in fact, that it is possible to drive the residual norm to zero, finding one of the solutions to the linear system, and have reconstructed images that are of poor subjective quality, perhaps worse than that of early iterates. Figure 12, for example, displays an exact image and two reconstructions of the data for that image. One reconstruction is made by computing $\mathbf{w} = B^{\dagger}\mathbf{f}$, while the other is made by running three PMLV cycles. The residual norms are 1.8×10^{-13} (pseudoinverse) and 4×10^{-4} (PMLV), but it can be seen that PMLV has produced a somewhat better reconstruction.

Such problems require some form of regularization to prevent the ill-posedness from completely corrupting the approximation. One way to do this [18] is to stop iterating when the algorithm begins to stall. An ad hoc approach to this is to measure the difference between successive residual norms, and stop iterating when a tolerance is achieved. A potentially more effective stopping criterion exists [20], based on a newly developed convergence theory for multilevel algorithms [7], but this is beyond the scope of interest here.

7. Spotlight image reconstruction. Often, one desires high resolution in a certain region of the image, for example, where a tumor is suspected. Discretizing the entire image space at a fine resolution may be impractical, as this leads to extremely large systems of equations. An attractive alternative is to discretize the region of special interest at a fine resolution and the remaining image space at a coarser resolution, leading to a *composite grid* problem. This is called *spotlighting* the region of interest. Numerous multigrid methods have been developed for handling composite grid problems [2], [16], [21]. One such method that we develop in the next section is a consequence of PML methodology.

7.1. Fast adaptive composite grid (FAC). The spotlight tomography problem is essentially a composite grid problem, in which an operator equation $Lu = f$ must be solved on a composite grid $\Omega^{\underline{h}}$ comprised of a global coarse grid Ω^{2h} and one or more local refinement grid Ω^h (the refinement grid may itself be a composite grid, which permits recursive refinement). Fast adaptive composite grid methods (FAC) were developed [14], [16] in order to utilize multigrid technology to treat such problems efficiently. It comes from the PML methodology by simply restricting the fine grid subspaces to local collections of detectors.

FAC succeeds because it handles the composite grid as a nested sequence of regular grids that can be treated independently using virtually any regular grid method. The key ingredients lie in having appropriate representations of the operator and intergrid transfer operators. Thus, grid functions u must be representable on the composite, global, and refinement grids ($u^{\underline{h}}$, u^{2h}, u^h), and operators must exist to transfer grid functions between these grids ($I_{\underline{h}}^h : \Omega^{\underline{h}} \to \Omega^h$, $I_{\underline{h}}^{2h} : \Omega^{\underline{h}} \to \Omega^{2h}$, $I_h^{\underline{h}} : \Omega^h \to \Omega^{\underline{h}}$, and $I_{2h}^{\underline{h}} : \Omega^{2h} \to \Omega^{\underline{h}}$). Finally, it is critical that the operator L be representable on these grids ($L^{\underline{h}}$, L^{2h}, L^h). In general, the details of these operators and representations can be developed in a straightforward fashion once the grids are defined; the details, however, are very technical and can be found in [12], [14], and [16]. Once the operators and grids are defined, FAC proceeds in the following two-step sequence:

Step 1: *Set* $f^{2h} \leftarrow I_{\underline{h}}^{2h}(f^{\underline{h}} - L^{\underline{h}}u^{\underline{h}}), \quad u^{2h} \leftarrow (L^{2h})^{-1} f^{2h}, \quad u^{\underline{h}} \leftarrow u^{\underline{h}} + I_{2h}^{\underline{h}} u^{2h}.$

Step 2: *Set* $f^{h} \leftarrow I_{\underline{h}}^{h}(f^{\underline{h}} - L^{\underline{h}}u^{\underline{h}}), \quad u^{h} \leftarrow (L^{h})^{-1} f^{h}, \quad u^{\underline{h}} \leftarrow u^{\underline{h}} + I_{h}^{\underline{h}} u^{h}.$

Despite this formal definition, FAC need not utilize exact solvers on the global coarse grid or the refinement patch. Historically, FAC has been used predominantly with iterative methods [14].

7.2. The spotlight grid. We utilize FAC methodology to devise a discretization for the spotlight tomography problem. We begin with a global grid Ω^{2h} generated by the natural pixels ψ_j^{2h}. We next add a refinement grid Ω^h by forming strips ψ_j^h. For each view we choose pairs of strips whose union conforms exactly to one of the global strips. As a very simple example, consider the discretization given by three views, each consisting of four strip pixels (Fig. 13, left). The resulting global (coarse grid) operator is the 12 × 12 matrix $B_{2h,2h}$, the matrix whose (j, k)th entry is $b_{jk}^{2h} = \langle \psi_j^{2h}, \psi_k^{2h} \rangle$. The refinement grid Ω^h is formed by partitioning one strip from each view into two strips. Hence ψ_2^{2h} generates ψ_1^h and ψ_2^h. Similarly, strips 3 and 4 on the refinement grid are a partition of ψ_7^{2h}, and ψ_{11}^{2h} generates refinement strips 5 and 6 (Fig. 13, right). The refinement strips ψ_k^h generate a refinement patch operator $B_{h,h}$, formed in the usual manner by $b_{jk}^h = \langle \psi_j^h, \psi_k^h \rangle$, giving a 6 × 6 matrix in the simple example. The composite grid $\Omega^{\underline{h}}$ formed in this manner is shown at the bottom of Fig. 13.

The basic assumption of the discretization, that the unknown image is a linear combination of the strip functions, is unchanged. The total number of strips is $N = N_{2h} + N_h$, the sum of the number of strips on the global grid and the refinement patch. Hence, the image $u = A^{\underline{h}*}\mathrm{w}$ is given by (3) where $\psi_j = \psi_j^{2h}$ if $1 \le j \le N_{2h}$ and $\psi_j = \psi_{j-N_{2h}}^h$ if $N_{2h} < j \le N$. We can define the composite grid version of $A^{\underline{h}}$ by $(Au)_j = \langle \psi_j, u \rangle$ using the same ordering of the ψ_j's. The discretized composite grid operator $B_{\underline{h},\underline{h}}$ may then be computed in the standard way, by $B_{\underline{h},\underline{h}} = A^{\underline{h}}A^{\underline{h}*}$, and it has, as its (j, k)th entry, the element $b_{jk}^{\underline{h}} = \langle \psi_j, \psi_k \rangle$. This method accounts for the interaction of the refinement grid with the global coarse grid, including the inner products between the fine strips and coarse strips. In fact, this leads to a natural

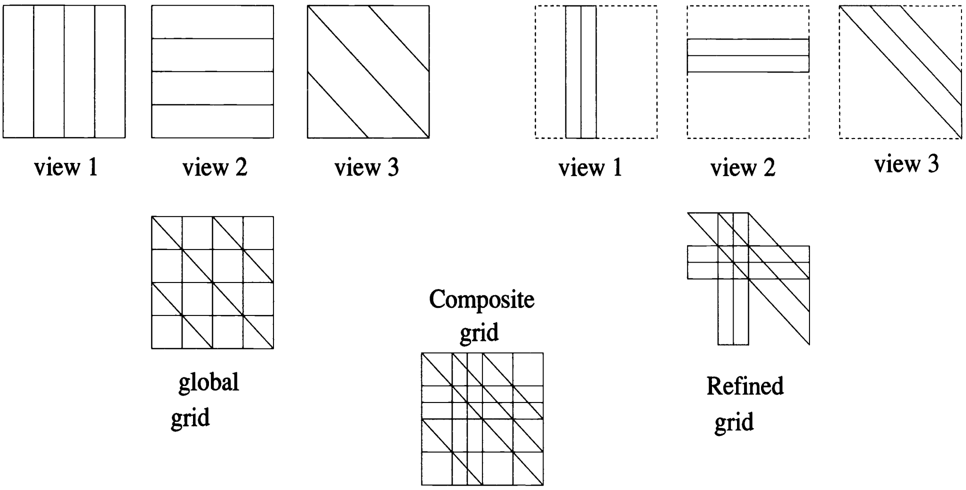


FIG. 13. *The global grid Ω^h (bottom left) is generated by strips from three views (top left), while the refinement patch (bottom right) is generated by finer strips from the three views (top right). The union of all strips generates the composite grid (bottom center).*

partitioning of the composite grid operator as

$$B_{\underline{h},\underline{h}} = \begin{pmatrix} B_{2h,2h} & B_{2h,h} \\ B_{h,2h} & B_{h,h} \end{pmatrix}.$$

We observe that by the ordering of the ψ_j's and the definition of $A^{\underline{h}*}$ we are led to a natural definition of the composite grid unknown, namely $u^{\underline{h}} = (\mathbf{w}^{2h} \quad \mathbf{w}^h)^T$. Naturally, we must have a compatible composite grid data vector $\mathbf{f}^{\underline{h}} = (\mathbf{f}^{2h} \quad \mathbf{f}^h)^T$. Conceivably, the coarse grid data and the refinement data could be acquired in separate recordings, but it is more likely that a single data set be generated, from which the coarse and refinement data are derived. The composite grid problem $B_{\underline{h},\underline{h}}\mathbf{w}^{\underline{h}} = \mathbf{f}^{\underline{h}}$ then becomes

$$\begin{pmatrix} B_{2h,2h} & B_{2h,h} \\ B_{h,2h} & B_{h,h} \end{pmatrix} \begin{pmatrix} \mathbf{w}^{2h} \\ \mathbf{w}^h \end{pmatrix} = \begin{pmatrix} \mathbf{f}^{2h} \\ \mathbf{f}^h \end{pmatrix}. \tag{11}$$

7.3. FAC implementation. It can be shown [14], [20] that FAC in this setting is equivalent to applying two steps of *block* Gauss–Seidel iteration to the system (11). That is, FAC takes the two-step form:

Step 1: *Set* $\mathbf{w}^{2h} \leftarrow B^{-1}_{2h,2h}(\mathbf{f}^{2h} - B_{2h,h}\mathbf{w}^h).$

Step 2: *Set* $\mathbf{w}^{h} \leftarrow B^{-1}_{h,h}(\mathbf{f}^{h} - B_{h,2h}\mathbf{w}^{2h}).$

These steps are formal, of course, since we know that $B_{2h,2h}$ is singular. In fact, we take $\mathbf{w} \leftarrow B^{-1}\mathbf{f}$ to mean "solve $B\mathbf{w} = \mathbf{f}$," which need not be done with exact solvers. In principle we may apply any method to these subproblems: ART, filtered backprojection, Fourier methods. A natural choice is an iterative method, such as Gauss–Seidel or multigrid. Noting that each of the steps are solving a "residual" equation on one of the grids, this process may be viewed as one of multilevel correction.

The composite grid operator $B_{\underline{h},\underline{h}}$ possesses a host of useful and interesting properties [20], related to a family of useful properties generated by the discretization method and inherent

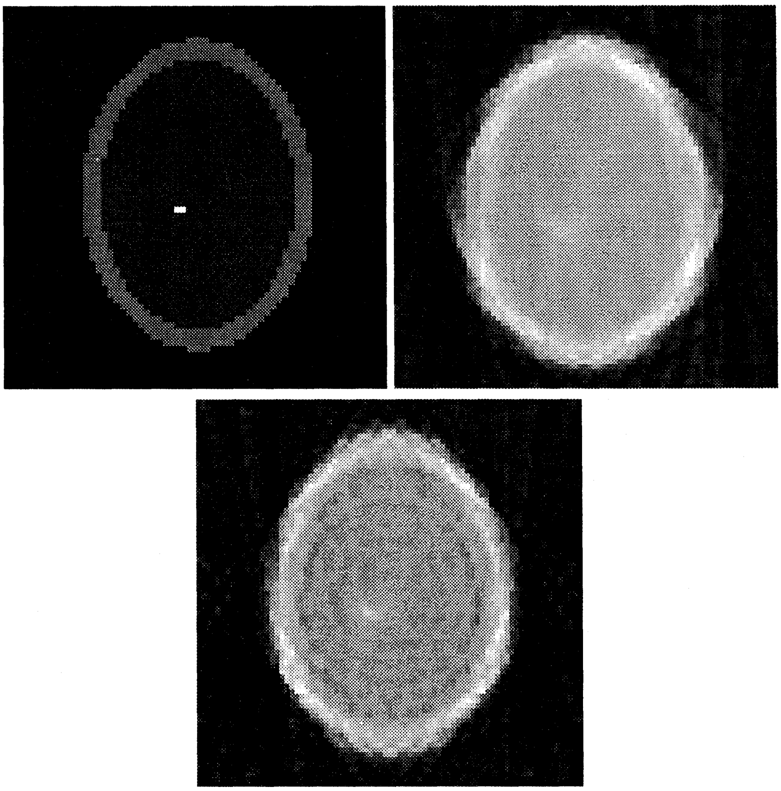


FIG. 14. *The "exact" image used to generate data for the spotlight tomography problem is shown on the top left. In the top right is shown the PML reconstruction on the global coarse grid* Ω^{2h}*, using data generated for* 20 *angles with* 32 *strips per angle. On the bottom is the spotlight reconstruction, generated using data for strips half the width of the global coarse grid, over the central region of the image.*

in the global and local operators $B_{2h,2h}$ and $B_{h,h}$. Space limitations do not permit elaboration here, nor is there room for a performance analysis. A rigorous treatment of the method, including a performance assessment, is forthcoming [8], [20].

We demonstrate the promise of the spotlight method with a simple example. A "brain" phantom is generated, consisting of a uniform grey region within the skull (high-density elliptical ring). Embedded in the grey region is a small square high-density region. Data is generated by integrating the product of this image with the characteristic functions of the strips representing a 20 angle, 32 bins-per-angle discretization. The square of high density has width equal to one half the width of the individual strips making up the global coarse grid. The "exact" image is shown on the left of Fig. 14, while the global coarse grid reconstruction is shown on the top right.

A single-level refinement region is generated by refining one half of the strips in the center of the set for each angle, using strips of half the width of those on the global coarse grid. The reconstructed image using the spotlight method with one cycle of FAC is shown on the bottom of Fig. 14. Both the global coarse grid and refinement grid portions of the composite grid were solved using three V-cycles, each with two relaxation sweeps on the downward leg and one relaxation sweep on the upward leg. The high-density region, which does not show in the global coarse grid reconstruction, appears in the spotlight reconstruction. This is demonstrated a bit more clearly in Fig. 15, in which only the central region of each of the reconstructions is shown.

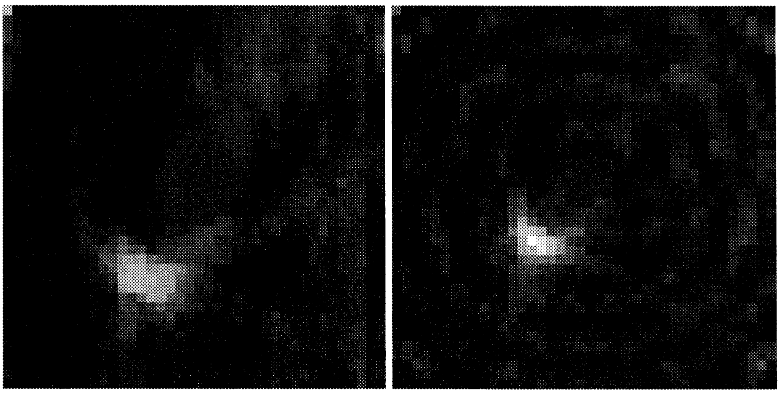


FIG. 15. *The central portions of global coarse grid (left) and spotlight (right) reconstructions of Fig. 14 are shown in this figure. The image reconstructed from the composite grid data clearly shows the object of interest, while the global coarse grid image fails to resolve it.*

We chose this example because of its clarity for the reader. However, it does not really illustrate the practical benefits of FAC because it costs essentially the same as would solving the globally refined problem. In practice, we envision that FAC will be used for spotlighting smaller features of the image, and to much finer detail. That is, rather than refining one half of the strips on each angle by splitting them once, we foresee refining a much smaller region, such as one tenth of the strips along each angle, to a resolution four or eight times that of the global coarse grid. In such settings the benefits of spotlighting would be very substantial.

8. Concluding remarks. The results presented here are encouraging, in that they demonstrate that multilevel methodology can be applied to the image reconstruction problem with some hope of success. The benefits of multilevel reconstruction, the way we have developed it, remain somewhat limited, although we do see images of quality equal to those produced by Gauss–Seidel, achieved at somewhat lower cost.

We believe that this limitation may stem from restricting the coarsening to be within projections. That is, we have reduced only the number of detectors per angle, not the number of angles themselves. Evidence gathered by examining the near null space components suggests that angle coarsening is essential for efficiency at coarse grain resolution, which multigrid methods always face. This is currently being explored.

The results presented here show great promise in the area of spotlight tomography, for cases where a finer resolution image is needed over portions of the image space. It is not feasible to compute entire images at the fine resolution, since such problems lead to extremely large, dense systems. As the simple example shows, however, PML can be used to formulate the spotlight problem to use FAC technology in a way that may lead to practical algorithms.

REFERENCES

[1] M. H. BOUNOCORE, W. R. BRODY, AND A. MACOVSKI, *A natural pixel decomposition for two dimensional image reconstruction*, IEEE Trans. Biomed. Engrg., BME-28 (1981), pp. 69–78.

[2] A. BRANDT, *Multi-level adaptive techniques (MLAT) for partial differential equations: Ideas and software*, in Proceedings of the Symposium on Mathematical Software, 1977, Academic Press, New York, 1977, pp. 277–318.

[3] ———, *Multi-level adaptive solutions to boundary value problems*, Math. Comp., 31 (1987), pp. 333–390.

[4] ———, *Rigorous local mode analysis of multigrid*, in Preliminary Proceedings of the Fourth Copper Mountain Conference on Multigrid Methods, J. Mandel and S. F. McCormick, eds., 1989.

[5] W. L. BRIGGS, *A Multigrid Tutorial*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1987.

[6] K. J. Cavanaugh and V. E. Henson, *A multilevel cost-space approach to solving the balanced long transportation problem*, in Proceedings of the 6th Copper Mountain Conference on Multigrid Methods, Vol. CP-3224, NASA Conference Publications, 1993.

[7] C. C. Douglas and J. Douglas, *A unified convergence theory for abstract multigrid or multilevel algorithms, serial and parallel*, SIAM J. Numer. Anal., 30 (1993), pp. 136–158.

[8] V. E. Henson, M. A. Limber, S. F. McCormick, and B. T. Robinson, *Multilevel projection methods for spotlight computed tomography*, in preparation.

[9] G. T. Herman, *Image Reconstruction from Projections*, Academic Press, Orlando, FL, 1980.

[10] M. A. Limber, T. A. Manteuffel, S. F. McCormick, and D. S. Sholl, *Optimal resolution in maximum entropy image reconstruction from projections with multigrid acceleration*, in Proceedings of the Sixth Annual Copper Mountain Conference on Multigrid Methods, 1993.

[11] A. K. Louis, *Medical imaging: The state of the art and future development*, Inverse Problems, 8 (1992), pp. 709–738.

[12] J. Mandel and S. F. McCormick, *Iterative solution of elliptic equations with refinement: The model multilevel case*, in Domain Decomposition Methods, T. F. Chan, ed., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1987, pp. 81–92.

[13] S. F. McCormick, *The methods of Kaczmarz and row orthogonalization for solving linear equations and least squares problems in Hilbert space*, Indiana Univ. Math. J., 26 (1977), pp. 1137–1150.

[14] ———, *Multilevel Adaptive Methods for Partial Differential Equations*, Frontiers in Applied Mathematics, Vol. 6, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1989.

[15] ———, *Multilevel Projection Methods for Partial Differential Equations*, CBMS-NSF Regional Conference Series in Applied Mathematics, Vol. 62, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.

[16] S. F. McCormick and J. Thomas, *The fast adaptive composite grid method (FAC) for elliptic boundary value problems*, Math. Comp., 46 (1986), pp. 439–456.

[17] S. F. McCormick and J. G. Wade, *Multilevel parameter estimation*, in Proceedings of the 5th Copper Mountain Conference on Multigrid Methods, 1991.

[18] F. Natterer, *The Mathematics of Computerized Tomography*, John Wiley and Sons, New York, 1986.

[19] T.-S. Pan, A. E. Yagle, N. H. Clinthorne, and W. L. Rogers, *Acceleration and filtering in the generalized Landweber iteration using a variable shaping matrix*, IEEE Trans. Medical Imaging, 12 (1993), pp. 278–286.

[20] B. T. Robinson, *A Multilevel Approach to the Algebraic Image Reconstruction Problem*, Ph.D. thesis, Naval Postgraduate School, Monterey, CA, 1994.

[21] U. Rüde, *Mathematical and Computational Techniques for Multilevel Adaptive Methods*, Frontiers in Applied Mathematics, Vol. 13, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1993.

[22] J. Ruge and K. Stüben, *Algebraic multigrid (AMG)*, in Multigrid Methods, S. F. McCormick, ed., Frontiers in Applied Mathematics, Vol. 3, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1987, pp. 131–177.

[23] L. A. Shepp and B. F. Logan, *The Fourier reconstruction of a head section*, IEEE Trans. Nucl. Sci., NS-21 (1974), pp. 21–43.

[24] H. Stark, J. W. Woods, I. Paul, and R. Hingorani, *Direct Fourier reconstruction in computer tomography*, IEEE Trans. Acoust., Speech Signal Process., ASSP-29 (1981), pp. 237–244.

[25] K. Tanabe, *Projection method for solving a singular system of linear equations and its applications*, Numer. Math., 17 (1971), pp. 203–214.

SIAM J. SCI. COMPUT.
Vol. 17, No. 1, pp. 217–226, January 1996

GMRES AND INTEGRAL OPERATORS*

C. T. KELLEY† AND Z. Q. XUE†

Abstract. In this paper we show how the properties of integral operators and their approximations are reflected in the performance of the GMRES iteration and how these properties can be used to smooth the GMRES iterates by an implicit application of Nyström interpolation, thereby strengthening the norm in which convergence takes place. The smoothed iteration has very similar properties to Broyden's method. We present an example to illustrate the ideas.

Key words. integral equations, GMRES iteration, Broyden's method

AMS subject classifications. 65F10, 65J10, 65R20

1. Introduction. In this paper we consider the performance of the GMRES [27] iteration for linear equations of the form

$$Au = u - Ku = f \tag{1.1}$$

with K a compact operator on a separable Hilbert space H. An example of such an operator is an integral operator on $H = L^2[0, 1]$ of the form

$$Ku(x) = \int_0^1 k(x, y)u(y)\, dy \tag{1.2}$$

where k is continuous.

Our setting is that of [15] where issues similar to those raised in this paper were considered in the context of Broyden's method [4] for linear and nonlinear equations. Broyden's method has also been considered as a linear equation solver in [5], [12], [13], [25], and [20]. Let H be a separable real Hilbert space and let $X \subset H$ be a Banach space such that the inner product $(\cdot, \cdot)$ in H is continuous from $X \times X \to R$. This implies that there is C_X such that

$$\|u\|_H \leq C_X \|u\|_X \tag{1.3}$$

for all $u \in X$. Let $K \in \mathcal{COM}(H, X)$, which is the space of compact operators from H to X. Of course, we may also regard K as an element of $\mathcal{COM}(H)$, which is the space of compact operators on H. In the context of the integral operator (1.2) with continuous k, $H = L^2[0, 1]$, $X = C[0, 1]$, and $C_X = 1$.

Algorithms such as GMRES and Broyden's method, which depend on notions of orthogonality, could use the Hilbert space inner product of H to solve equations in which the right-hand side $f \in X$. However, a convergence theory based entirely on a Hilbert space formulation would show that the resulting sequence is convergent in the topology of H but not necessarily in that of the Banach space X in which the problem may have been originally posed. Hence, we face an apparent conflict between the topology in which the problem was posed and the inner product (and hence Hilbert space) nature of the algorithm. This issue was resolved in [15] in the context of Broyden's method. For the linear equations context of this paper the result of [15] is that the Broyden iterations converge q-superlinearly in the topology of X provided $K \in \mathcal{COM}(H, X)$ and $f \in X$.

*Received by the editors May 4, 1994; accepted for publication (in revised form) March 24, 1995.

† Center for Research in Scientific Computation and Department of Mathematics, North Carolina State University, P.O. Box 8205, Raleigh, NC 27695-8205 (Tim_Kelley@ncsu.edu, xue@math.ncsu.edu). This research was supported by National Science Foundation grants DMS-9024622 and DMS-9321938. Computing activity was also partially supported by an allocation of time from the North Carolina Supercomputing Center.

Nyström interpolation [24], i.e., replacing an approximate solution u by $\bar{u} = f + Ku$, can be used to create approximations accurate in stronger norms than the original [17] to improve the overall accuracy of a discrete approximation [28] and as a method for smoothing an intergrid transfer in multilevel schemes [1], [3], [14]. We have the first purpose in mind here. Following GMRES with a Nyström interpolation would, as we shall see, result in accuracy in the X-norm. The cost, however, would be an additional application of K.

The purpose of this paper is to show how GMRES can be modified to incorporate Nyström interpolation at a very small cost in both computational effort and algorithmic complexity. The result is an algorithm that has the simultaneous X and H convergence property of Broyden's method. The example we give in §4 compares GMRES, both with and without Nyström interpolation, and Broyden's method as primary solvers and not in the context of a multilevel method, where they could be either used as coarse mesh solvers or as the primary solver in a nested iteration approach. The reasons for this are that the independence of our results to mesh size can best be illustrated if the solvers perform the same task for each mesh. The merits of our implicit Nyström interpolation carry over to the multilevel situation, enabling coarse mesh solvers to be accurate in the appropriate discrete X-norm, which is important for the convergence theory.

Throughout this paper we assume that A is a nonsingular linear operator on H and X. We consider convergence rate estimates of the form

$$\|r_k\|_H \leq \tau_k \|r_0\|_H, \tag{1.4}$$

where $r_k = f - Au_k$, and the sequence of real numbers $\{\tau_k\}$ converges to zero and is independent of the right-hand side f of (1.1).

Rates of convergence of the form (1.4) can be derived from resolvent integration [21], [22] for any K such that $I - K$ has bounded inverse and 1 is in the unbounded component of the resolvent set of K. If K is compact, more precise information can be obtained. In fact the GMRES iterates converge r-superlinearly to the solution in a way that is independent of the right-hand side. This means that the sequence $\{\tau_k\}$ converges q-superlinearly to zero; i.e.,

$$\lim_{k\to\infty} \frac{\tau_{k+1}}{\tau_k} = 0.$$

In the case of normal or diagonalizable (similar to normal) compact operators a q-superlinearly convergent sequence $\{\tau_k\}$ can be directly related to spectral properties of K in a very simple way. In [18] assumptions on the rate of decay of the spectrum were used for this. In order to illustrate how the smoothing properties of K might influence the convergence rate, below we present a slight extension of the result in [18]. While this result follows from the general theory in [22], we believe that its direct and brief proof is worth inclusion. We denote by $\kappa_H(A)$ the H-norm condition number of A;

$$\kappa_H(A) = \|A\|_H \|A^{-1}\|_H.$$

THEOREM 1.1. *Let H be a separable Hilbert space, $K \in \mathcal{COM}(H)$, let $A = I - K$ be nonsingular, and let S be a nonsingular bounded operator on H such that*

$$L = SKS^{-1}$$

is normal. Let $\{\lambda_i\}$ be the eigenvalues of A ordered so that

$$|\lambda_i - 1| \geq |\lambda_{i+1} - 1| \quad \textit{for } i \geq 1; \tag{1.5}$$

then for all $k \geq 1$ the GMRES residuals r_k satisfy

$$\|r_k\| \leq \kappa_H(S)\|r_0\|2^k \prod_{i=1}^{k} |1 - \lambda_i^{-1}|. \tag{1.6}$$

Proof. For $k = 1, \ldots$ define

$$p_k(z) = \prod_{i=1}^{k}(1 - \lambda_i^{-1}z). \tag{1.7}$$

Since $p_k(0) = 1$ for all k we have [27]

$$\|r_k\|_H \leq \kappa_H(S)\|r_0\|_H \sup_m |p_k(\lambda_m)|. \tag{1.8}$$

For k fixed, $p_k(\lambda_m) = 0$ for $1 \leq m \leq k$. For $m > k$,

$$\begin{aligned} |p_k(\lambda_m)| &\leq \prod_{i=1}^{k} |1 - \lambda_i^{-1}\lambda_m| = \prod_{i=1}^{k} |\lambda_i^{-1}||\lambda_i - \lambda_m| \\ &\leq \prod_{i=1}^{k} |\lambda_i^{-1}|(|\lambda_i - 1| + |1 - \lambda_m|) \leq \prod_{i=1}^{k} |\lambda_i^{-1}|2|\lambda_i - 1| \\ &= 2^k \prod_{i=1}^{k} |1 - \lambda_i^{-1}|. \end{aligned}$$

This completes the proof. □

Since $\lambda_i \to 1$, the sequence

$$\tau_k = 2^k \prod_{i=1}^{k} |1 - \lambda_i^{-1}|$$

is q-superlinearly convergent. If, say, K is normal (so $S = I$) and the eigenfunctions of K are smooth, then the rate of convergence of λ_i to one reflects both the smoothness of the kernel k and the convergence rate of the sequence $\{\tau_k\}$.

2. Convergence in a stronger norm.

In this section we show how, given a rate estimate like (1.4) for the sequence of residuals, the GMRES iteration can be modified to produce a sequence that converges with the same rate in the norm of X. This will lead to a modified form of GMRES that uses information at hand to perform Nyström interpolation implicitly without the need for an additional application of the operator K. Proposition 2.1 is a coupling of the convergence rate derived above with the standard smoothing result for Nyström interpolation [2]. We provide the simple proof to help clarify the ideas.

PROPOSITION 2.1. *Let $\{u_k\}$ be the sequence of GMRES iterates. Assume that* (1.4) *holds for some sequence $\{\tau_k\}$. Let $\bar{u}_k = u_k + r_k$. Then*

$$\|\bar{u}_k - u^*\|_X \leq \|K\|_{\mathcal{L}(H,X)}\kappa_H(A)\tau_k\|u_0 - u^*\|_X. \tag{2.1}$$

Proof. First note that (1.4) implies that

$$\|u_k - u^*\|_H \leq \tau_k\kappa_H(A)\|u_0 - u^*\|_H. \tag{2.2}$$

Since

$$\bar{u}_k = u_k + r_k = f + K u_k,$$

continuity of K as a map from H to X implies that

$$\|\bar{u}_k - u^*\|_X = \|K(u_k - u^*)\|_X \le \|K\|_{\mathcal{L}(H,X)} \|u_k - u^*\|_H.$$

This completes the proof. □

The interpolant $\bar{u}_k$ is as easy to compute as u_k upon exit from the main loop in GMRES. An algorithmic description of GMRES follows.

ALGORITHM 2.1. *Algorithm* gmres(u, f, A, ϵ)

```
1. $r = f - Au$, $v_1 = r/\|r\|_H$, $\rho = \|r\|_H$, $\beta = \rho$, $k = 1$
2. While $\rho > \epsilon \|b\|_H$ do
   (a) $v_{k+1} = Av_k$
       for $j = 1, \dots, k$
         i. $h_{jk} = (v_{k+1}, v_j)$
         ii. $v_{k+1} = v_{k+1} - h_{jk} v_j$
   (b) $h_{k+1,k} = \|v_{k+1}\|_H$
   (c) $v_{k+1} = v_{k+1}/\|v_{k+1}\|_H$
   (d) $e_1 = (1, 0, \dots, 0)^T \in R^{k+1}$
       Minimize $\|\beta e_1 - H_k y\|_{R^{k+1}}$ to obtain $y \in R^k$
   (e) $\rho = \|\beta e_1 - H_k y\|_{R^{k+1}}$
   (f) $k = k + 1$
3. $u_k = u_0 + V_k y$
```

In step 3 $V_k : R^k \to H$ is defined by

$$V_k y = \sum_{j=1}^{k} y_j v_j.$$

We can use the fact that

$$r_k = f - Au_k = V_{k+1}(\beta e_1 + H_k y) = r_0 + V_{k+1} H_k y$$

to recover $\bar{u}_k$ with no additional operator–vector products involving A. In fact if $z = H_k y$ and we define a vector

$$w = (w_1, w_2, \dots, w_{k+1})^T \in R^{k+1}$$

by $w_i = z_i + y_i$ for $1 \le i \le k$ and $w_{k+1} = z_{k+1}$, we have

$$\bar{u}_k = \bar{u}_0 + V_{k+1} w, \tag{2.3}$$

which can simply replace the computation of u_k in step 3 of gmres. We will refer to the resulting algorithm as *smoothed GMRES*. Note that smoothed GMRES differs from GMRES only in the final output, where (2.3) is used to compute the final result.

As an algorithm for the solution of linear compact fixed point problems, smoothed GMRES shares two properties with Broyden's method. Both converge superlinearly to the solution in the topologies of both H and X and both require storage of the iteration history. Older implementations of Broyden's method require storage of two vectors for each iteration, [10], [23], [7]. However recent work for linear [9] and nonlinear [16] problems show how to use only a single vector, making the two algorithms competitive.

3. Discrete problems. While we state and prove our results in the infinite-dimensional setting they apply equally well when K is an approximation of some compact operator and X and H are both R^N for some N with different norms, say, discrete approximations of L^2 (for H) and L^∞ or C^l (for X) norms.

To illustrate this point, consider the integral operator defined by (1.2). A standard approximation by an N-point quadrature rule would lead to the discrete problem

$$u_i - \sum_{j=1}^{N} k(x_i, x_j) w_j = f(x_i), \tag{3.1}$$

where $\{x_i\}$ and $\{w_i\}$ are the nodes and weights of the quadrature rule. Assume that f and k are l times continuously differentiable. Define H to be R^N with the inner product

$$(u, v)_H = \sum_{j=1}^{N} u_j v_j w_j$$

and associated norm

$$\|u\|_H = \left(\sum_{j=1}^{N} u_j^2 w_j \right)^{1/2}.$$

We give $C^l[0, 1]$ the norm

$$\|u\|_{C^l} = \|u\|_\infty + \|d^l u/dx^l\|_\infty.$$

If we replace $\|d^l u/dx^l\|_\infty$ with a lth divided difference we may define an approximate C^l norm on R^N. If X is R^N with such an approximate C^l norm, then the matrix

$$K_{ij} = k(x_i, x_j) w_j$$

satisfies, with C independent of N

$$\|Ku\|_X \le C \|u\|_H,$$

and the development in the above section may be applied for each level of discretization N with the constants in the estimates being independent of N.

For example, if $l = 1$, and the nodes of quadrature rule are $\{ih\}_{i=0}^{N-1}$ with $h = 1/(N-1)$, we may define the divided difference $D_1 u \in R^{N-1}$ by

$$(D_1 u)_i = (u_{i+1} - u_i)/h \tag{3.2}$$

and the X-norm by

$$\|u\|_X = \|u\|_\infty + \|D_1 u\|_\infty.$$

In §4 we present an example showing how smoothed GMRES can capture the smoothing properties of the Broyden iteration that are missed by GMRES alone.

4. Example. As an example we consider the source iteration operator from linear transport theory [19], [26]. We will describe the equation and its discretization only briefly and refer to [19] and [26] for more details. We consider the equation

$$\mu \frac{\partial \psi}{\partial x}(x, \mu) + \psi(x, \mu) = \frac{c(x)}{2} \int_{-1}^{1} \psi(x, \mu') \, d\mu' \tag{4.1}$$

for $x \in (0, \tau)$ with boundary conditions

$$\psi(0, \mu) = F_l(\mu), \quad \mu > 0; \qquad \psi(\tau, \mu) = F_r(\mu), \quad \mu < 0. \tag{4.2}$$

In (4.1) and (4.2) the intensity ψ is the unknown real valued function of x and μ. $\tau < \infty$, $c \in C([0, \tau])$, and F_l and F_r are given continuous real valued functions of μ. It is known [6] that the boundary value problem (4.1)–(4.2) has a unique solution if $0 \leq c(x) < 1$.

The intensity can be computed directly once the flux

$$f(x) = \frac{1}{2}\int_{-1}^{1} \psi(x, \mu')\, d\mu' \tag{4.3}$$

is known. By integrating forward ($\mu > 0$) and backward ($\mu < 0$) in x we derive the integral equation for f,

$$f(x) - \int_0^{\tau} k(x, y) f(y)\, dy = g(x) \tag{4.4}$$

where

$$k(x, y) = \frac{1}{2} E_1(|x - y|) c(y),$$

$$E_1(|x - y|) = \int_0^1 \exp\left(\frac{-|x - y|}{\nu}\right) \frac{d\nu}{\nu},$$

and

$$g(x) = \frac{1}{2}\int_0^1 \exp\left(\frac{-x}{\nu}\right) F_l(\nu)\, d\nu$$
$$+\frac{1}{2}\int_0^1 \exp\left(\frac{-(\tau - x)}{\nu}\right) F_r(-\nu)\, d\nu.$$

It is known [6], [8] that the integral operator K in (4.4) has spectral radius < 1 and hence (4.4) has a unique solution. It is also easy to see from the formula for k that K maps L^2 into any of the spaces

$$X_\epsilon^l = C[0, 1] \cap C^l[\epsilon, 1].$$

By examining the formula for g we see that the solution $f \in C[0, 1]$ is infinitely differentiable in compact subsets of $(0, 1]$ and hence lies in any of the spaces X_ϵ^l.

We discretize (4.4) with the standard *discrete ordinates* approximation to (4.1), which is not a direct approximation of (4.4) by a quadrature rule at all. We approximate the integral in (4.1) by a 20-point double Gaussian quadrature rule (10-point Gaussian quadratures on each of the intervals $(-1, 0)$ and $(0, 1)$) in line with the analysis in [26] and then integrate forward and backward with the Crank–Nicolson scheme on a spatial mesh with N points. The convergence theory developed in [19] and [26] implies that the discrete form K of the integral operator satisfies

$$\|Ku\|_{l,\epsilon} \leq M(l, \epsilon)\|u\|_H$$

for some $M(k, \epsilon)$, which is independent of the spatial mesh. Here $\|\cdot\|_H$ denotes the scaled Euclidean norm

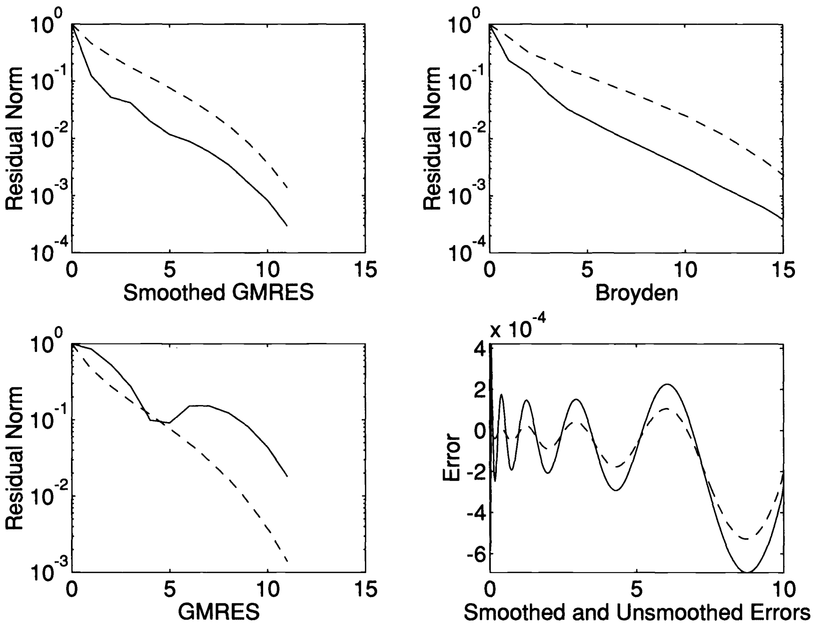


FIG. 4.1. $N = 401$.

$$\|u\|_H = \left(\frac{1}{N}\sum_{j=1}^{N} u_j^2\right)^{1/2},$$

and the discrete X_ϵ^l norms are given by

$$\|u\|_{l,\epsilon} = \|u\|_\infty + \|D_1^l u_\epsilon\|_\infty. \tag{4.5}$$

In (4.5) the operator D_1 is defined by (3.2) and u_ϵ is the vector defined by

$$(u_\epsilon)_i = \begin{cases} 0, & ih \le \epsilon, \\ u_i, & ih > \epsilon. \end{cases}$$

We set $F_l(\mu) = 1$, $F_r(\mu) = 0$ for all μ, and $\tau = 10$. We set $c(x) = \exp(-x/100)$, which is a special case of the class considered in [11]. We performed three sets of computations for $N = 401$, $N = 801$, and $N = 1601$. We used the initial iterate $u_0 = 0$ and solved the equation with GMRES, smoothed GMRES, and Broyden's method. We terminated the iteration when the discrete H-norm residual had been reduced by a factor of $5/(N-1)^2$. We used the solution with $N = 6401$ (terminated when the H-norm residual had been reduced by a factor of 10^{-12} as a representation of the exact solution). We computed the discrete X_ϵ^l-norms with $\epsilon = .025$.

For $N = 401$ (Fig. 4.1), $N = 801$ (Fig. 4.2), and $N = 1601$ (Fig. 4.3) we plot relative residual norms as functions of the iteration number in both the discrete X_ϵ^1-norm (solid line) and the discrete H-norm (dashed line). In each figure there are plots of residual histories for both GMRES and smoothed GMRES (SGMRES). In those plots the dashed line is the H-norm relative residual for both GMRES and SGMRES. The solid line is the relative residual norm in X_ϵ^1, which we computed directly in GMRES by explicitly computing the residual and its X_ϵ^1-norm or by using (2.3). We also plot the error in the final result as a function of $x \in [0, \tau]$ for smoothed (dashed line) and unsmoothed (solid line) GMRES. One can clearly see the

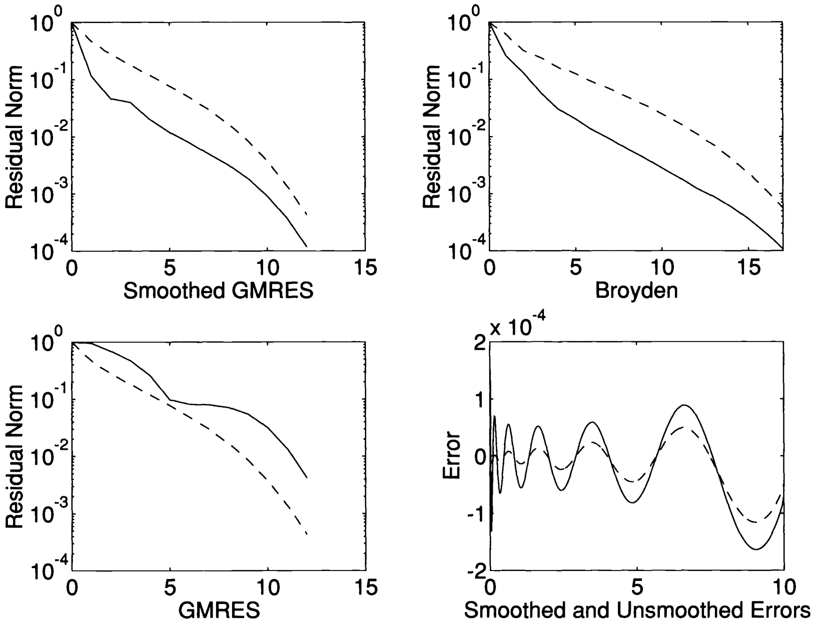


FIG. 4.2. $N = 801$.

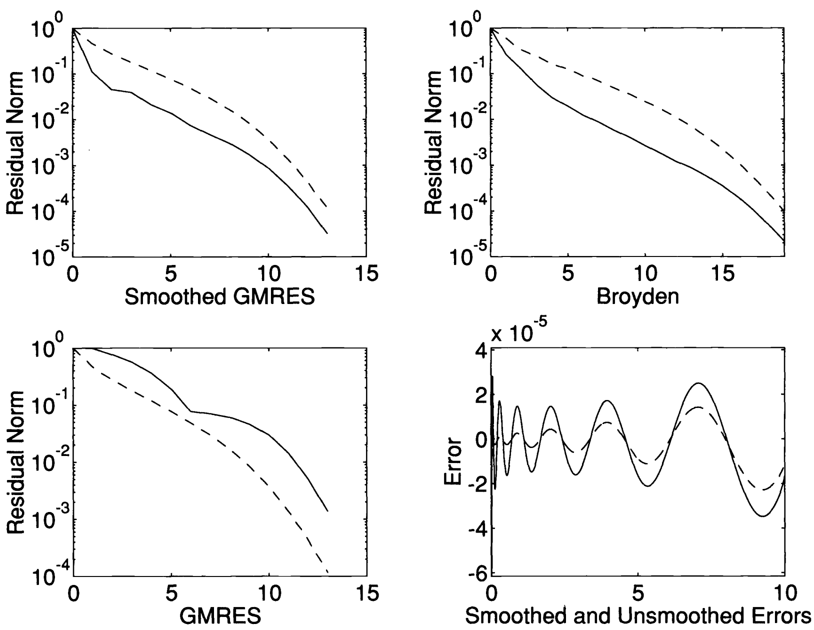


FIG. 4.3. $N = 1601$.

effects of the smoothing especially near $x = 0$ where the errors of the unsmoothed solution oscillate much more strongly.

To show how the smoothing affects the results in other norms, we present in Table 4.1 the norms of the errors in the discrete L^∞, X_ϵ^1, X_ϵ^2, and X_ϵ^3 norms for GMRES and SGMRES. One can see from these tables that the effect of the Nyström interpolation becomes more dramatic as the differentiability in the norms increases.

TABLE 4.1
Errors in various norms.

	$N = 401$		$N = 801$		$N = 1601$	
Norm	GMRES	SGMRES	GMRES	SGMRES	GMRES	SGMRES
H	3.1865e-4	2.3279e-4	7.6028e-5	4.8816e-5	1.7326e-5	9.8239e-6
L^∞	6.9223e-4	5.2749e-4	1.6349e-4	1.1584e-4	6.1573e-5	2.2828e-5
X_ϵ^1	4.0814e-2	8.0632e-3	7.0348e-3	1.2897e-3	3.1247e-3	3.9660e-4
X_ϵ^2	1.9878e+0	2.4950e-1	5.3146e-1	3.0577e-2	2.2838e-1	1.1983e-2
X_ϵ^3	8.3956e+1	7.5656e+0	2.5227e+1	1.6119e+0	9.2401e+0	8.5306e-1

The computations illustrate how the smoothed GMRES iteration gives better performance in the X_ϵ^1-norm. The curves for the various values of m are quite similar, indicating that the infinite-dimensional analysis can be observed numerically. Note also that the X_ϵ^1 and H relative residuals for the Broyden iteration, which requires more work than GMRES, are very close, in line with the theory in [15].

The tables and figures were created with MATLAB version 4.0a on a Sun SPARC 20 model 61 workstation running SUN OS 4.1.3.

Acknowledgments. The authors thank Anne Greenbaum for pointing out [22] and Andreas Griewank for telling us about [9].

REFERENCES

[1] K. E. ATKINSON, *Iterative variants of the Nyström method for the numerical solution of integral equations*, Numer. Math., 22 (1973), pp. 17–31.

[2] ———, *A Survey of Numerical Methods for Fredholm Integral Equations of the Second Kind*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1976.

[3] H. BRAKHAGE, *Über die numerische Behandlung von Integralgleichungen nach der Quadraturformelmethode*, Numer. Math., 2 (1960), pp. 183–196.

[4] C. G. BROYDEN, *A class of methods for solving nonlinear simultaneous equations*, Math. Comp., 19 (1965), pp. 577–593.

[5] W. BURMEISTER, *Zur Konvergenz einiger verfahren der konjugierten Richtungen*, in Proc. Internationaler Kongreß über Anwendung der Mathematik in dem Ingenieurwissenschaften, Weimar, 1975.

[6] I. W. BUSBRIDGE, *The Mathematics of Radiative Transfer*, Cambridge Tracts No. 50, Cambridge University Press, Cambridge, 1960.

[7] R. H. BYRD, J. NOCEDAL, AND R. B. SCHNABEL, *Representation of quasi-Newton matrices and their use in limited memory methods*, Math. Programming, 63 (1994), pp. 129–156.

[8] S. CHANDRASEKHAR, *Radiative Transfer*, Dover, New York, 1960.

[9] P. DEUFLHARD, R. W. FREUND, AND A. WALTER, *Fast secant methods for the iterative solution of large nonsymmetric linear systems*, Impact Comput. Sci. Engrg., 2 (1990), pp. 244–276.

[10] M. ENGELMAN, G. STRANG, AND K. J. BATHE, *The application of quasi-Newton methods in fluid mechanics*, Internat. J. Numer. Methods Engrg., 17 (1981), pp. 707–718.

[11] R. GARCIA AND C. SIEWERT, *Radiative transfer in finite inhomogeneous plane-parallel atmospheres*, J. Quant. Spectrosc. Radiat. Transfer, 27 (1982), pp. 141–148.

[12] D. M. GAY, *Some convergence properties of Broyden's method*, SIAM J. Numer. Anal., 16 (1979), pp. 623–630.

[13] A. GRIEWANK, *On the iterative solution of differential and integral equations using secant updating techniques*, in The State of the Art in Numerical Analysis, A. Iserles and M. J. D. Powell, eds., Clarendon Press, Oxford, 1987, pp. 299–324.

[14] W. HACKBUSCH, *Multigrid methods of the second kind*, in Multigrid Methods for Integral and Differential Equations, Oxford University Press, Oxford, 1985.

[15] D. M. HWANG AND C. T. KELLEY, *Convergence of Broyden's method in Banach spaces*, SIAM J. Optim., 2 (1992), pp. 505–532.

[16] C. T. KELLEY, *Iterative Methods for Linear and Nonlinear Equations*, SIAM Frontiers in Applied Mathematics, Vol. 16, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1995.

[17] C. T. KELLEY AND E. W. SACHS, *Broyden's method for approximate solution of nonlinear integral equations*, J. Integral Equations Appl., 9 (1985), pp. 25–44.

[18] T. Kerkhoven and Y. Saad, *On acceleration methods for coupled nonlinear elliptic systems*, Numer. Math., 60 (1992), pp. 525–548.

[19] E. W. Larsen and P. Nelson, *Finite difference approximations and superconvergence for the discrete ordinate equations in slab geometry*, SIAM J. Numer. Anal., 19 (1982), pp. 334–348.

[20] J. J. Moré and J. A. Trangenstein, *On the global convergence of Broyden's method*, Math. Comp., 30 (1976), pp. 523–540.

[21] N. M. Nachtigal, S. C. Reddy, and L. N. Trefethen, *How fast are nonsymmetric matrix iterations?*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 778–795.

[22] O. Nevanlinna, *Convergence of Iterations for Linear Equations*, Birkhäuser, Basel, 1993.

[23] J. Nocedal, *Theory of algorithms for unconstrained optimization*, Acta Numerica, 1 (1991), pp. 199–242.

[24] E. J. Nyström, *Über die praktische Auflösung von Integralgleichungen mit Anwendungen auf Randwertaufgaben*, Acta Math., 54 (1930), pp. 185–204.

[25] D. P. O'Leary, *Why Broyden's nonsymmetric method terminates on linear equations*, SIAM J. Optim., 5 (1995), pp. 231–235.

[26] J. Pitkäranta and R. Scott, *Error estimates for the combined spatial and angular approximations of the transport equation in slab geometry*, SIAM J. Numer. Anal., 20 (1983), pp. 922–950.

[27] Y. Saad and M. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[28] I. H. Sloan, *Improvement by iteration for compact operator equations*, Math. Comp., 30 (1976), pp. 758–764.

ITERATIVE METHODS FOR TOTAL VARIATION DENOISING*

C. R. VOGEL† AND M. E. OMAN‡

Abstract. Total variation (TV) methods are very effective for recovering "blocky," possibly discontinuous, images from noisy data. A fixed point algorithm for minimizing a TV-penalized least squares functional is presented and compared with existing minimization schemes. A variant of the cell-centered finite difference multigrid method of Ewing and Shen is implemented for solving the (large, sparse) linear subproblems. Numerical results are presented for one- and two-dimensional examples; in particular, the algorithm is applied to actual data obtained from confocal microscopy.

Key words. total variation, denoising, image reconstruction, multigrid methods, confocal microscopy, fixed point iteration

AMS subject classifications. 65F10, 65N55

1. Introduction. The problem of denoising, or estimating an underlying function from error-contaminated observations, occurs in a number of important applications, particularly in probability density estimation and image reconstruction. Consider the model equation

$$z = u + \epsilon, \tag{1.1}$$

where u represents the desired true solution, ϵ represents error, and z represents the observed data. A number of approaches can be taken to estimate u. These include spline smoothing (see [18]), filtering using Fourier and wavelet transforms, and total variation (TV)-based denoising. Figure 1 illustrates the qualitative differences between these various approaches on a simple one-dimensional test case. It is not the goal of this paper to carry out an exhaustive comparison of TV with standard denoising methods. For that, see [15] and the analysis in [5]. Suffice it to say that TV denoising is extremely effective for recovering "blocky," possibly discontinuous, functions from noisy data. It *is* the goal of this paper to present a new algorithm for TV denoising and to compare it with some existing TV-based methods.

In their seminal paper on TV denoising, Rudin, Osher, and Fatemi [15] considered the constrained minimization problem

$$\min_u \int_\Omega |\nabla u|\, dx \quad \text{subject to} \quad \|u - z\|^2 = \sigma^2, \tag{1.2}$$

where the parameter σ describes the magnitude of the error ϵ in the data in the model equation (1.1). Here Ω is a bounded, convex region in d-dimensional space, $|\cdot|$ denotes the Euclidean norm in R^d, and $\|\cdot\|$ denotes the norm on $L^2(\Omega)$.

Here we will consider a closely related problem—the minimization of the TV-penalized least squares functional

$$f(u) = \frac{1}{2}\|u - z\|^2 + \alpha\, J_\beta(u), \tag{1.3}$$

where

$$J_\beta(u) = \int_\Omega \sqrt{|\nabla u|^2 + \beta^2}\, dx \tag{1.4}$$

*Received by the editors May 23, 1994; accepted for publication (in revised form) March 20, 1995.

†Department of Mathematical Sciences, Montana State University, Bozeman, MT 59717 (vogel@math.montana.edu). Research was supported in part by the NSF under grant DMS-9303222.

‡Scalable Computing Laboratory, Ames Laboratory (USDOE), Iowa State University, Ames, IA 50011 (oman@scl.ameslab.gov). Research was supported in part by a DOE-EPSCoR graduate fellowship at Montana State University.

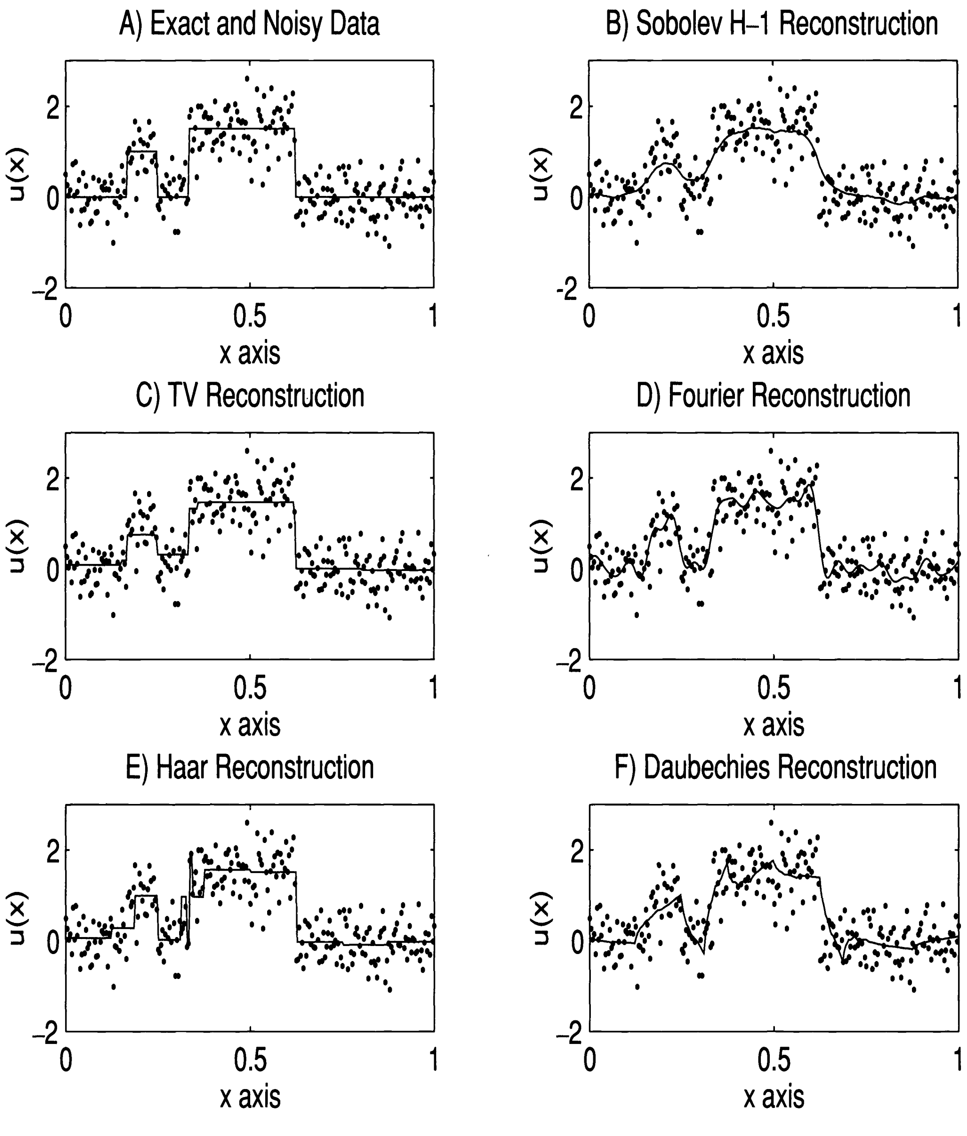


FIG. 1. *Denoised reconstructions obtained using a variety of filtering techniques. Dotted lines represent noisy data. Solid line in subplot A is the exact solution. Solid lines in subplots B–F are reconstructions. Reconstructions E and F were obtained using Haar and Daubechies's wavelets, respectively. See* §4 *for details.*

and α and β are (typically small) positive parameters. The parameter α controls the trade-off between goodness of fit to the data, as measured by $\|u - z\|$, and the variability of the solution, measured by $J_\beta(u)$. When $\beta = 1$, $J_\beta(u)$ represents the surface area of the graph of u, while $\beta = 0$ gives the total variation of u.

When $\beta = 0$, TV-penalized least squares can be viewed as a penalty method (see [11]) to solve the constrained problem (1.2). The penalty parameter α in (1.3) is inversely proportional to the Lagrange multiplier for (1.2). This penalty approach is standard in the inverse problems community and is commonly referred to as Tikhonov regularization. Provided the parameters are selected appropriately, the solutions obtained by these two methods are identical. However, from a computational standpoint, unconstrained problems are much easier to implement than constrained problems.

To solve their constrained minimization problem, Rudin, Osher, and Fatemi applied artificial time evolution. In the context of the unconstrained problem (1.3), this amounts to assuming u is a function of time t (as well as space) and then time-integrating the differential equation

$$\frac{\partial u}{\partial t} = -g(u), \quad t > 0, \tag{1.5}$$

$$u = u^{(0)}, \quad t = 0, \tag{1.6}$$

to steady state. Here $g(u)$ denotes the gradient (derivative with respect to u) of the TV-penalized least squares functional (1.3), and $u^{(0)}$ is an initial guess for the solution. Formally, g is a nonlinear elliptic partial differential operator with homogeneous Neumann boundary conditions

$$g(u) = u - \alpha \nabla \cdot \left(\frac{\nabla u}{\sqrt{|\nabla u|^2 + \beta^2}} \right) - z, \quad x \in \Omega, \tag{1.7}$$

$$\frac{\partial u}{\partial n} = 0, \quad x \in \partial\Omega. \tag{1.8}$$

After spatial discretization, Rudin, Osher, and Fatemi applied explicit (forward Euler) time marching to obtain a gradient descent scheme. In the context of (1.5), this approach yields

$$u^{(k+1)} = u^{(k)} - \tau_k \, g(u^{(k)}), \quad k = 0, 1, \ldots. \tag{1.9}$$

A line search (see [4]) can be added to select the step size τ_k in a manner that gives sufficient decrease in the objective functional in (1.3) to guarantee convergence to a minimizer. This gives the method of steepest descent (see [11]). While numerical implementation is straightforward, steepest descent has rather undesirable asymptotic convergence properties that can make it very inefficient. Obviously, one can apply other standard unconstrained optimization methods with better convergence properties, like the nonlinear conjugate gradient method or Newton's method. These methods converge rapidly near a minimizer *provided the objective functional depends smoothly on u.* When $\beta = 0$, the term J_β in (1.3) is not differentiable. For small values of β, the near nondifferentiability of the objective functional results in a loss of robustness and efficiency for higher-order methods like Newton's method.

In this paper we introduce an alternative approach to minimizing (1.3) that we call "lagged diffusivity fixed point iteration," denoted by FP. At a minimizer, $g(u) = 0$ or, equivalently,

$$u + \alpha \, L(u) u = z, \tag{1.10}$$

where $L(u)$ is the diffusion operator whose action on a function v is given by

$$L(u)v = -\nabla \cdot \left(\frac{1}{\sqrt{|\nabla u|^2 + \beta^2}} \nabla v \right). \tag{1.11}$$

FP iteration can be expressed as

$$\left(1 + \alpha \, L(u^{(k)})\right) u^{(k+1)} = z, \quad k = 0, 1, \ldots. \tag{1.12}$$

Note that at each iteration one must solve a linear diffusion equation, whose diffusivity depends on the previous iterate $u^{(k)}$, to obtain the new iterate $u^{(k+1)}$. In our numerical experiments, we have observed global convergence of FP iteration. We suspect this is true in general because the mapping $u \mapsto L(u)u$ is monotone (see [9]). Hence, there appears to be no need for a "globalization" procedure like a line search to guarantee convergence, as is the case with standard optimization methods. In addition, this method exhibits rapid linear convergence for a broad range of the parameters α and β.

In the following section, we discuss the mathematical structure of the TV-penalized least squares functional (1.3) and the equations that arise in its minimization. Section 3 deals with numerical implementation issues like discretization, stopping criteria, and iterative methods to solve (large, sparse) linear systems. In the final section we present a numerical comparison of FP iteration with Newton's method and steepest descent. Results of a numerical study of the effects of various parameters (e.g., α and β in (1.3)) are also presented in this section. Finally, we apply FP iteration to denoise actual data obtained from a confocal scanning microscope [19].

2. Mathematical structure. In this section we discuss the mathematical structure of the TV-penalized least squares functional (1.3) and the implications of this structure for numerical methods. An analysis of a similar functional,

$$\frac{1}{2}\|Ku - z\|^2 + \alpha J_\beta(u), \tag{2.1}$$

has been carried out in [1], where K is a compact linear operator mapping $L^p(\Omega)$ into $L^2(\Omega)$, with $1 \le p < d/(d-1)$. d is the spatial dimension. This analysis relies on the relative compactness of sets of the form

$$\{u \in L^p(\Omega) : \|u\|_p + J_\beta(u) \le B\}, \tag{2.2}$$

where B is a fixed constant, in $L^p(\Omega)$.

Perhaps the most important difference in the analysis of (1.3) is the fact that sets of the form (2.2) are *not* relatively compact in $L^2(\Omega)$ for dimensions $d > 1$. Following §2 of [1], define the functional

$$Q(u, \vec{v}) = \int_\Omega \left[\frac{1}{2}(u-z)^2 - \alpha\, u\, \nabla \cdot \vec{v} + \alpha\beta\sqrt{1 - |\vec{v}|^2}\right] dx. \tag{2.3}$$

Also define

$$f(u) = \sup_{\vec{v} \in \mathcal{V}} Q(u, \vec{v}), \tag{2.4}$$

where

$$\mathcal{V} = \{\vec{v} \in C_0^\infty(\Omega; R^d) : |\vec{v}(x)| \le 1\}. \tag{2.5}$$

For u sufficiently smooth (say, in $C^1(\Omega)$), this coincides with the functional in (1.3)–(1.4). We wish to find $u^* \in L^2(\Omega)$ for which

$$f(u^*) = \inf_{u \in L^2(\Omega)} f(u). \tag{2.6}$$

THEOREM 2.1. *Problem* (2.6) *has a unique solution.*

Proof. Note that $f(u)$ is L^2-coercive; i.e., $f(u) \to \infty$ whenever $\|u\|_2 \to \infty$. This combined with the weak compactness of closed balls in $L^2(\Omega)$ and the weak lower-semicontinuity of f yields existence. Uniqueness follows from the strict convexity of the L^2 norm. □

While we have not yet been able to rigorously establish this fact, it appears that the minimizer is stable with respect to perturbations in the data z and the parameters α and β. This has the following implications:

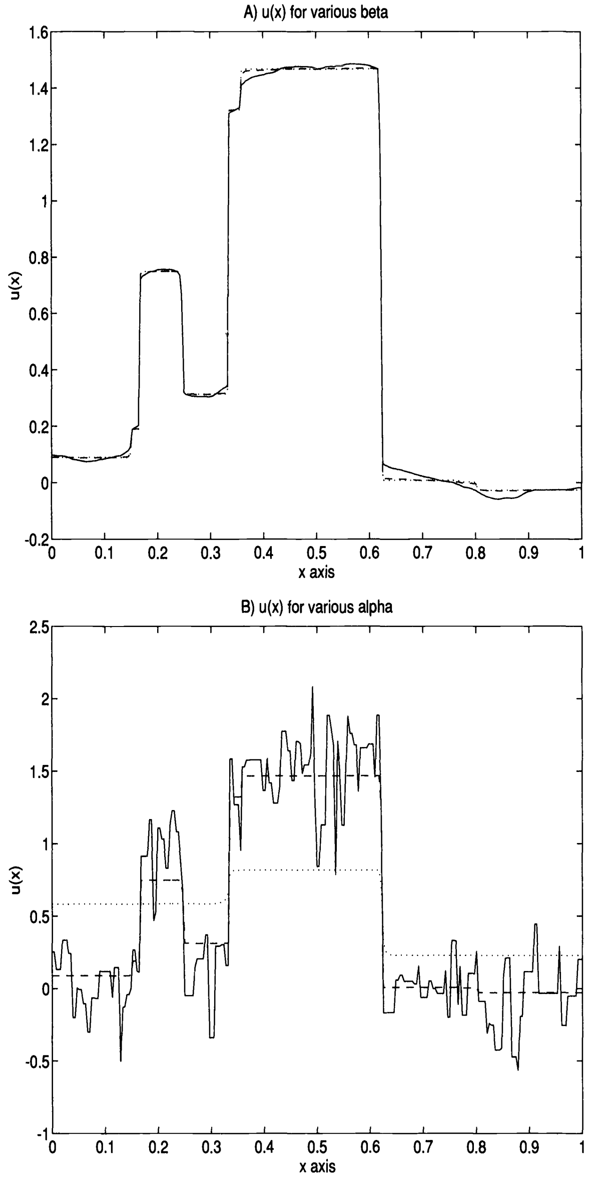


FIG. 2. *Subplot A shows TV reconstructions for various β's with fixed α = 0.01. Solid line corresponds to $\beta = .1$, dashed line to $\beta = .01$, and dotted line to $\beta = 10^{-6}$. Subplot B shows reconstructions for various α's with fixed $\beta = 10^{-4}$. Solid line corresponds to $\alpha = .001$, dashed line to $\alpha = .01$, and dotted line to $\alpha = .1$. See* §4 *for details.*

(i) Taking β small but positive in (1.3) gives minimizers that are close (in an L^2 sense) to minimizers obtained with $\beta = 0$. This is illustrated in Fig. 2A.

(ii) As $\alpha \rightarrow 0$, the minimizer $u = u_\alpha$ in (1.3) tends to the (noisy) data z. On the other hand, as α becomes large, u_α tends to the mean value of z. This is illustrated in Fig. 2B.

We next examine the operators arising in the minimization of (1.3). Taking the first Gateaux derivative in the direction v and assuming u and v are smooth, one obtains the gradient

$$\langle g(u), v\rangle = \int_\Omega \left((u-z)v + \alpha\left(\frac{\nabla u \cdot \nabla v}{\sqrt{|\nabla u|^2+\beta^2}}\right)\right) dx. \tag{2.7}$$

Similarly, one obtains the Hessian

$$\langle H(u)v, w\rangle = \int_\Omega \left(vw + \alpha\frac{\nabla v \cdot \nabla w}{(|\nabla u|^2+\beta^2)^{3/2}}\right) dx. \tag{2.8}$$

The quadratic approximation

$$f(u+s) = f(u) + \langle g(u), s\rangle + \frac{1}{2}\langle H(u)s, s\rangle + o(\|s\|^2) \tag{2.9}$$

is the basis for the analysis of standard optimization methods. In particular, the spectrum of the Hessian $H(u)$ determines the asymptotic convergence rate of the methods of steepest descent and the (nonlinear) conjugate gradient method (see [11]). One obtains local quadratic convergence for Newton's method assuming that $H(u)$ is Lipschitz continuous and has a bounded inverse (see [4]). The size of the convergence region is proportional to the inverse of the Lipschitz constant and the bound on the inverse. Note that if $\beta = 0$ and ∇u vanishes anywhere in Ω, then $f(u)$ is not differentiable. This is a consequence of the nondifferentiability of the Euclidean norm. This difficulty is overcome when $\beta > 0$, but the Lipschitz constant behaves like β^{-3} when β is small. This explains the deterioration in the performance of Newton's method observed in §4 for small values of β.

Again assuming sufficient smoothness and applying Green's identity (integration by parts) in (2.7), one obtains

$$\begin{aligned}\langle g(u), v\rangle &= \int_\Omega \left((u-z) - \alpha\nabla\cdot\left(\frac{\nabla u}{\sqrt{|\nabla u|^2+\beta^2}}\right)\right) v\, dx \\ &+\alpha\int_{\partial\Omega}\frac{\nabla u\cdot n}{\sqrt{|\nabla u|^2+\beta^2}} ds,\end{aligned} \tag{2.10}$$

where n is the outward unit normal to the boundary $\partial\Omega$. Consequently, a minimizer for (1.3) is a weak solution to the nonlinear second-order elliptic PDE

$$g(u) \stackrel{\text{def}}{=} u - \alpha\nabla\cdot\left(\frac{\nabla u}{\sqrt{|\nabla u|^2+\beta^2}}\right) - z = 0, \quad x \in \Omega, \tag{2.11}$$

$$\frac{\partial u}{\partial n} = 0, \quad x \in \partial\Omega. \tag{2.12}$$

This can be expressed in operator form

$$A(u)u = (1+\alpha L_\beta(u))u = z, \tag{2.13}$$

where

$$L_\beta(u)v = -\nabla\cdot\left(\kappa_\beta(u)\nabla v\right) \tag{2.14}$$

and

$$\kappa_\beta(u) = \frac{1}{\sqrt{|\nabla u|^2+\beta^2}}. \tag{2.15}$$

Note that the operator L_β is symmetric and positive semidefinite. Its null space consists of the constant functions. The diffusivity $\kappa_\beta(u)$ is bounded above by $\frac{1}{\beta}$ and below by zero.

Finally, observe that the Hessian is related to the operators $A(u)$ and $L_\beta(u)$ by

$$\begin{aligned} H(u) &= 1 + \alpha L_\beta(u) + \alpha L'_\beta(u)u \\ &= A(u) + \alpha L'_\beta(u)u, \end{aligned} \tag{2.16}$$

where $'$ denotes differentiation with respect to u.

3. Numerical implementation. Any discretization of (1.3) should allow for very sharp gradients without introducing spurious oscillations in the solution u. We have implemented several finite element methods (which are based on (2.7)) with piecewise linear basis functions and the first-order accurate finite difference scheme described in [15] (based on (2.11)–(2.12)). The resulting discrete systems are quite similar. These schemes yield discrete Hessians H, cf. (2.8), and matrix operators A, cf. (2.13), which are symmetric and positive definite (SPD) and sparse—tridiagonal in one space dimension and block tridiagonal in two dimensions. Let $\mathbf{u}$ and $\mathbf{z}$ denote the vectors (mesh functions) obtained from the discretization of u and z, respectively. Also, denote by $f(\mathbf{u})$, $\mathbf{g}(\mathbf{u})$, $H(\mathbf{u})$, and $A(\mathbf{u})$, respectively, the discretization of the objective functional (1.3), its gradient, Hessian, and the operator A in (2.13).

What follows is a generic algorithm for the minimization of $f(\mathbf{u})$. The superscript (k) denotes iteration count. Let $\mathbf{u}^{(0)}$ be an initial guess. For $k = 0, 1, \ldots,$

1. Compute a descent direction $\mathbf{d}^{(k)}$ for f at $\mathbf{u}^{(k)}$.
2. $\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + \lambda^* \mathbf{d}^{(k)}$, where

$$\lambda^* = \text{argmin}_{\lambda>0} f(\mathbf{u}^{(k)} + \lambda \mathbf{d}^{(k)}).$$

3. Check stopping criteria.

For the method of steepest descent, $\mathbf{d}_{SD}^{(k)} = -\mathbf{g}^{(k)} \overset{\text{def}}{=} -\mathbf{g}(\mathbf{u}^{(k)})$, while for Newton's method,

$$\mathbf{d}_N^{(k)} = -H(\mathbf{u}^{(k)})^{-1}\mathbf{g}^{(k)}. \tag{3.1}$$

Note that the line search in step 2 may be replaced by a trust region method [4]. Either "globalization" technique will guarantee convergence to the minimizer of f.

For our FP iteration, steps 1 and 2 are combined. One obtains $\mathbf{u}^{(k+1)}$ directly by solving the linear system

$$A(\mathbf{u}^{(k)})\mathbf{u}^{(k+1)} = \mathbf{z}. \tag{3.2}$$

Setting $\mathbf{d}_{FP}^{(k)} = \mathbf{u}^{(k+1)} - \mathbf{u}^{(k)}$ yields

$$\begin{aligned} \mathbf{d}_{FP}^{(k)} &= -A(\mathbf{u}^{(k)})^{-1}\left(A(\mathbf{u}^{(k)})\,\mathbf{u}^{(k)} - z\right) && (3.3)\\ &= -A(\mathbf{u}^{(k)})^{-1}\, g(\mathbf{u}^{(k)}). && (3.4) \end{aligned}$$

The second equality follows from (2.11) and (2.14). Hence FP iteration is of quasi-Newton form, and existing convergence theory in [4] can be applied. Since the matrix $A(\mathbf{u})$ is SPD with its minimum eigenvalue bounded away from zero, each of the $\mathbf{d}_{FP}^{(k)}$'s is a descent direction, and global convergence can be guaranteed by "globalization," i.e., appropriate step size control. In our computational experiments, we have not found globalization to be necessary. Comparing (3.1) and (3.3) and observing that the term $\alpha L'_\beta(u)u$ in (2.16) does not vanish, the asymptotic convergence rate is linear.

The following stopping criteria are standard (see [4]). δ_1, δ_2, and δ_3 are user-defined tolerances, $k_{\max}$ is an iteration limit, and $\|\cdot\|$ denotes the ℓ^2 norm.

$$\|\mathbf{u}^{(k+1)} - \mathbf{u}^{(k)}\| \leq \delta_1, \tag{3.5}$$

$$\|\mathbf{g}^{(k+1)})\| \leq \delta_2, \tag{3.6}$$

$$f(\mathbf{u}^{(k)}) - f(\mathbf{u}^{(k+1)}) \leq \delta_3 \, \|\mathbf{u}^{(k+1)} - \mathbf{u}^{(k)}\|, \tag{3.7}$$

$$k \geq k_{\max}. \tag{3.8}$$

Solving the linear systems. With both Newton's method (cf. (3.1)) and FP iteration (cf. (3.2)), one must solve a sparse SPD linear system at each iteration. In one space dimension, these systems are tridiagonal and can be solved directly in $\mathcal{O}(n)$ operations, where n is the order of the system. In two space dimensions, these systems are block tridiagonal. Direct banded system solvers, which have complexity $\mathcal{O}(n^{3/2})$, may be applied. One may also apply a variety of iterative methods, like preconditioned conjugate gradient (PCG) methods (see [2], [10]) and multigrid methods (see [12] and the references therein).

For certain linear SPD systems arising in the discretization of elliptic PDEs, one can achieve $\mathcal{O}(n)$ complexity with multigrid methods [12]. Our early multigrid implementations, which were based on standard finite difference or finite element discretizations and standard intergrid transfer operators, yielded very disappointing results when u was not smooth. This seems to be due to properties of the diffusion coefficient $\kappa_\beta(u)$; cf. equation (2.15). For nonsmooth u of bounded variation, $\kappa_\beta(u)$ is not smooth. Moreover, on the set (having Lebesgue measure zero) where u is discontinuous, $\kappa_\beta(u)$ vanishes.

To overcome these difficulties, we employed a cell-centered finite difference (CCFD) discretization (see [7] and the references therein). To solve the resulting linear systems, we applied a variant of the multigrid algorithm developed by Ewing and Shen [8]. While pure multigrid iterations were considered in [8], we applied a PCG iteration with a CCFD-based multigrid method as a preconditioner. See [17] for implementation details. Numerical results appear in the following section.

4. Numerical results. In this section we first present a numerical comparison of FP iteration, Newton's method, and steepest descent applied to minimize the TV-penalized least squares functional (1.3). Results are also presented that illustrate the effects of varying the parameters α and β on these methods.

First consider the one-dimensional test problem of denoising the data presented in Fig. 1. The exact (noise-free) solution is

$$u_{\text{exact}}(x) = \chi_{[1/6,1/4]} + \frac{3}{2}\chi_{[1/3,5/8]}, \tag{4.1}$$

where $\chi_{[a,b]}$ denotes the indicator function for the interval $a \leq x \leq b$. The data was generated by evaluating u_{exact} at $N = 257$ equispaced points in the interval $0 \leq x \leq 1$ and adding pseudorandom, uncorrelated error (so-called "discrete white noise") $\{\epsilon_i\}_{i=1}^N$ having a Gaussian distribution with mean 0 and variance σ^2 selected so the noise-to-signal ratio

$$\sqrt{\frac{E(\sum_{i=1}^N \epsilon_i^2)}{\sum_{i=1}^N u(x_i)^2}} = 0.5. \tag{4.2}$$

Here $E(\cdot)$ denotes mathematical expectation. Subplot C shows the minimizer of theTV-penalized least squares functional (1.3). Subplot B shows the minimizer of a related H^1-penalized least squares functional, which is commonly used in data smoothing (see [18])

$$\|u - z\|^2 + \lambda \int_\Omega |\nabla u|^2 \, dx. \tag{4.3}$$

Here the penalty term is the square of the Sobolev H^1 seminorm. It does not allow discontinuous minimizers. On the other hand, it is easy to compute minimizers and is appropriate for denoising smooth functions. Subplot D was obtained by Fourier transforming the data, applying a low pass filter, and then applying the inverse transform. Subplots E and F were created with the aid of the software package `wavethresh`, as documented by Nason and Silverman [14]. Each of these reconstructions was obtained by applying a wavelet transform to the data, applying the universal filter of Donoho and Johnstone [6], and then applying an inverse wavelet transform. In Subplot E, Haar wavelets are used in the transformations (see [16] or [13] for a discussion of Haar wavelets). These wavelets are generated by a discontinuous mother wavelet and are of regularity level 1 (see [14]). This reconstruction clearly maintains the discontinuities of the true image; however, there appear to be extraneous effects similar to ringing that are not a part of the original image. Subplot F uses Daubechies's "extremal phase" wavelets (see [3]), which have regularity level 2 (see [14]). Here, both smoothing and ringing effects are apparent. In all cases, the filter parameters were selected so that

$$\sum_{i=1}^{N} (u(x_i) - z_i)^2 / N \approx \sigma^2. \tag{4.4}$$

Figure 2 shows the qualitative effects of varying the parameters α and β on the minimizer of the TV-penalized least squares functional (1.3). In subplot A, the fixed $\alpha = 0.01$ is selected so (4.4) is satisfied, and β is varied. Larger values of β have the effect of "rounding off sharp edges" in the reconstructions. In subplot B, $\beta = 10^{-2}$ is fixed and α is varied. Solutions tend to be piecewise constant. Larger values of α have the effect of reducing the number of piecewise constant regions.

Figure 3 illustrates the convergence behavior of the various methods for minimizing (1.3), as measured in the ℓ_1 norm of the gradient. In each case, the initial guess was taken to be the zero solution; i.e., $\mathbf{u}^{(0)} = 0$, and $\alpha = 0.01$. Subplots A and B also show the effect of varying the parameter β on the performance of Newton's method with a line search and FP iteration. Note that the Newton iteration converges rapidly for relatively large values of β. However, as β decreases, the performance decreases markedly. The line search restricts the size of steps in order to maintain a steady decrease in $f(\mathbf{u}^{(k)})$, but quadratic convergence is not attained until a very large number of iterations have been performed. See the discussion in §2 for an explanation. FP performance also drops off as β decreases, but unlike Newton's method, the drop off is gradual and there is no dramatic change as β becomes very small. Subplot C illustrates performance of the method of steepest descent. There is a substantial decrease in the norm of the gradient in the first few iterations, but after that the decrease is extremely slow. A thousand steepest descent iterations were required to obtain reconstructions comparable to those obtained with four or five Newton or FP iterations.

Figure 4 shows FP performance as measured by the objective functional (1.3). Note that $f(\mathbf{u}^{(k)})$ decreases monotonically.

Finally, we present a two-dimensional example from confocal microscopy (see [19]). The images in subplots A and B of Fig. 5 show rod-shaped bacteria on a stainless steel surface. The vertical axis represents recorded light intensity, while the horizontal axes represent scaled

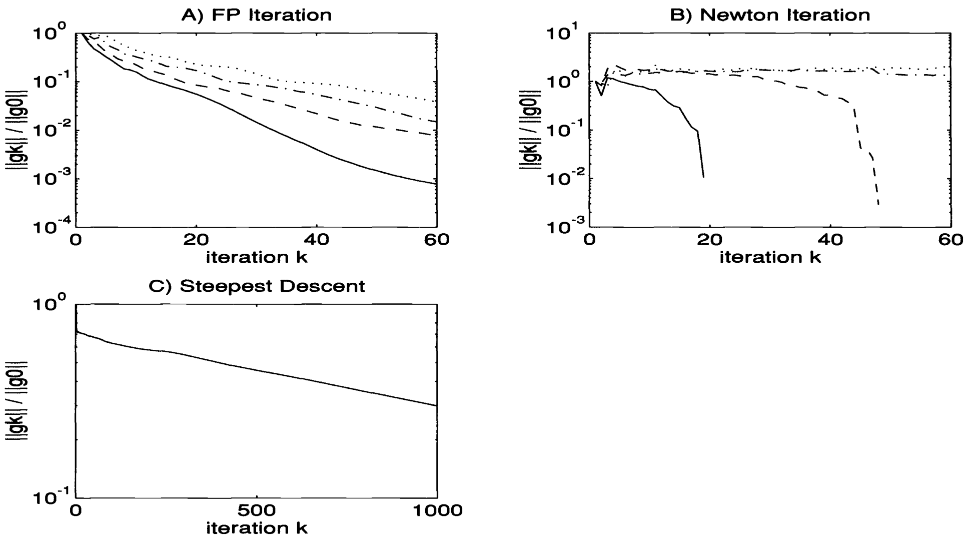


FIG. 3. *Performance of methods measured by the scaled ℓ_1 norm of the gradient $\|\mathbf{g}^{(k)}\|_1/\|\mathbf{g}^{(0)}\|_1$. $\alpha = 0.01$ is fixed throughout. Subplot C shows steepest descent performance for $\beta = 1$ only. Subplot A shows FP performance for $\beta = 1$ (solid line), $\beta = 0.1$ (dashed line), $\beta = 10^{-2}$ (dashed-dotted line), and $\beta = 10^{-3}$ (dotted line). Subplot B shows Newton performance for $\beta = 1$ (solid line), $\beta = 0.1$ (dashed line), $\beta = 10^{-2}$ (dashed-dotted line), and $\beta = 10^{-3}$ (dotted line).*

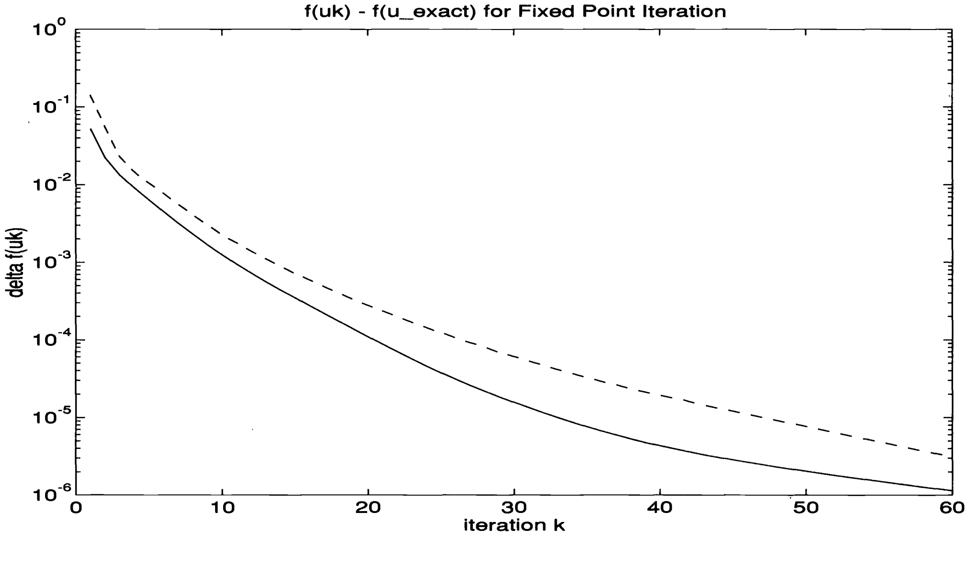


FIG. 4. *Performance of fixed point iteration measured by $f(\mathbf{u}^{(k)}) - f(\mathbf{u}^*)$, where $\mathbf{u}^*$ is the minimizer of f. Solid line is for $\beta = 1$ and dashed line is for $\beta = 0.1$. α is fixed at* 0.01.

pixel locations on a 64 × 64 grid. Figure 5A is an actual image recorded with a scanning confocal microscope. Figure 5B shows a TV reconstruction obtained from the FP algorithm with our CCFD multigrid PCG method used to solve the linear systems at each fixed point iteration. The actual computations were performed on a 256 × 256 pixel grid, with $n \approx 65,000$ unknowns. So that fine details are not obscured, only the upper left hand 64 × 64 subgrid is actually displayed. Parameter values are $\alpha = \beta = .01$.

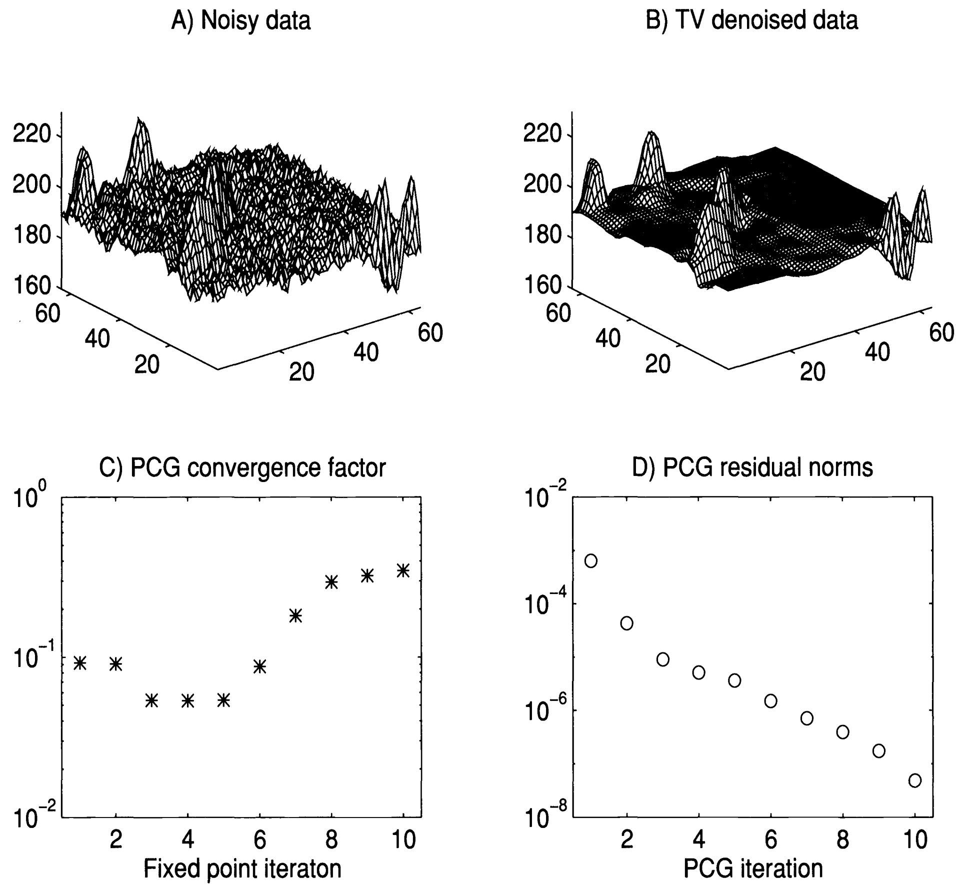


FIG. 5. *Subplot A shows a scanning confocal microscope image of rod-shaped bacteria on a stainless steel surface. Subplot B shows a TV reconstruction obtained using FP iteration. Subplots C and D illustrate performance of the linear system solver.*

Subplots C and D describe the performance of the CCFD multigrid PCG linear system solver. Subplot D shows the norms of residuals $\mathbf{r}_j = \mathbf{z} - A\mathbf{u}_j$ as a function of PCG iteration count at FP iteration 10. $A = A(\mathbf{u}^{(10)})$ is fixed throughout for this subplot. Define the PCG convergence factor to be the ratio $\rho_j = \|\mathbf{r}_j\|/\|\mathbf{r}_{j-1}\|$. Subplot C shows the geometric mean of the convergence factors, $\bar{\rho} = \exp(\sum_j^m \ln \rho_j/m)$, as a function of FP iteration count.

Finally, we note that far fewer that 10 fixed point iterations and 10 PCG iterations per FP iteration were required to obtain comparable denoised images. The purpose of the large number of iterations was to demonstrate the asymptotic convergence properties of the linear iterative method.

REFERENCES

[1] R. ACAR AND C. R. VOGEL, *Analysis of total variation penalty methods for ill-posed problems*, Inverse Problems, 10 (1994), pp. 1217–1229.

[2] O. AXELSSON AND V.A. BARKER, *Finite Element Solution of Boundary Value Problems*, Academic Press, New York, 1984.

[3] I. DAUBECHIES, *Orthonormal bases of compactly supported wavelets*, Comm. Pure Appl. Math., 41 (1988), pp. 909–996.

[4] J. E. Dennis, Jr. and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice–Hall, Englewood Cliffs, NJ, 1983.

[5] D. Dobson and F. Santosa, *Recovery of Blocky Images from Noisy and Blurred Data*, Tech. Report 94-7, Center for the Mathematics of Waves, University of Delaware, Newark, DE, 1994.

[6] D. L. Donoho and I. M. Johnstone, *Ideal spatial adaptation by wavelet shrinkage*, Biometrika, 81 (1994), pp. 425–455.

[7] R. E. Ewing and J. Shen, *A Discretization Scheme and Error Estimate for Second-Order Elliptic Problems with Discontinuous Coefficients*, Institute for Scientific Computation, Texas A & M University, College Station, preprint.

[8] ———, *A multigrid algorithm for the cell-centered finite difference scheme*, in Proc. 6th Copper Mountain Conference on Multigrid Methods, April 1993, NASA Conference Publication 3224.

[9] S. Fucik and A. Kufner, *Nonlinear Differential Equations*, Elsevier, New York, 1980.

[10] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 2nd ed., The Johns Hopkins University Press, Baltimore, MD, 1989.

[11] D. Luenberger, *Introduction to Linear and Nonlinear Programming*, Addison–Wesley, Reading, MA, 1965.

[12] S. F. McCormick, ed., *Multigrid Methods*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1987.

[13] Y. Meyer, *Wavelets: Algorithms and Applications*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1993.

[14] G. P. Nason and B. W. Silverman, *The discrete wavelet transform in S*, J. Comput. Graphical Statistics, 3 (1994), pp. 163–191.

[15] L. I. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Phys. D, 60 (1992), pp. 259–268.

[16] G. Strang, *Wavelet transforms versus Fourier transforms*, Amer. Math. Soc. Bull., 28 (1993), pp. 288–305.

[17] C. R. Vogel, *A Multigrid Method for Total Variation-Based Image Denoising*, in Control and Computation IV, K. Bowers and J. Lund, eds., Progress in Systems and Control Theory, Volume 20, Birkhäuser, Boston, 1995, pp. 323–331.

[18] G. Wahba, *Spline Models for Observational Data*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1990.

[19] T. Wilson, ed., *Confocal Microscopy*, Academic Press, New York, 1990.

SIAM J. SCI. COMPUT.
Vol. 17, No. 1, pp. 239–259, January 1996

016

MIGRATION OF VECTORIZED ITERATIVE SOLVERS TO DISTRIBUTED-MEMORY ARCHITECTURES*

CLAUDE POMMERELL† AND ROLAND RÜHL‡

Abstract. Distributed-memory parallel processors (DMPPs) can deliver peak performance higher than vector supercomputers while promising a better cost-performance ratio. Programming, however, is harder than on traditional vector systems, especially when problems necessitating unstructured solution methods are considered. A class of such applications, with large resource requirements, is the numerical solution of partial differential equations (PDEs) on nonuniformly refined three-dimensional finite element discretizations. Porting an application of this class from vector and shared-memory parallel machines to DMPPs involves some fundamental algorithm changes, such as grid decomposition, mapping, and coloring strategies. In addition, no standardized language interface is available to ease the efficient parallelization and porting among DMPPs and between vector computers and DMPPs.

This article describes how PILS—an existing package for the iterative solution of large unstructured sparse linear systems of equations on vector computers—was ported to DMPPs, using the parallelizing Fortran compiler Oxygen. Two DMPPs, namely an Intel Paragon and a Fujitsu AP1000, were used to evaluate the performance of the generated parallel program quantitatively. The results indicate how an application should be designed to be portable among supercomputers of different architecture. Several language and architecture features are essential for such a porting process and ease the parallelization of similar applications drastically.

Key words. distributed-memory parallel processing, large-scale scientific computing, iterative solvers, parallelizing compilers, inspector-executor, unstructured sparse matrix computations

AMS subject classifications. 65Y05, 68N20, 65F10

1. Introduction.

Recent development in VLSI technology has made distributed-memory parallel processors (DMPPs) attractive alternatives to vector supercomputers. DMPPs offer high peak performance for relatively low investment in computer hardware: powerful microprocessors can be used to build a DMPP processing element (PE) with only few integrated circuits. By connecting many such PEs together in a communication network, it is easy to achieve at least the same peak performance as featured by today's most powerful vector supercomputers.

Unfortunately, DMPPs are still much less comfortable to program. Programming comfort cannot be quantified as precisely as peak performance, but it is clear that more user efforts are required to implement efficient and portable programs. While early research in scientific parallel processing concentrated on isolated parallel algorithms and specialized architectures, many current projects aim to mitigate this lack of general programmability. Approaches include new software technology—like parallel algorithms, operating systems (OS) and parallelizing compilers—and new hardware designs, in particular more powerful DMPP interconnection networks. Portability among DMPPs and, even more, portability between DMPPs and vector supercomputers is often ignored.

In this article, we trace a way to extend the portability of a software package from vector computers to DMPPs, in the hope of providing a guideline for similar ports. Our project is a step toward the full parallelization of existing applications that are based on large and unstructured two- and three-dimensional finite element discretizations.

The article is organized as follows. After defining the project goal and rising the associated questions more precisely, we will introduce the hardware platforms and the two software packages PILS [29] and Oxygen [32] used in our study. We will then discuss additions to the

*Received by the editors May 19, 1994; accepted for publication (in revised form) December 23, 1994.

† AT&T Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974-0636. Current address: ABB Corporate Research, CH-5405 Baden-Dättwil, Switzerland (claude.pommerell@chcrc.abb.ch).

‡ ISE Integrated Systems Engineering AG, Technopark Zurich, Technoparkstr.1, Ch-8005 Zurich, Switzerland (ruehl@ise.ch).

numerical library PILS that enable an efficient parallelization with the parallelizing compiler Oxygen and conclude with a summary of our measurements on two selected DMPPs.

2. Goal of this project. The distinction between the two classes of supercomputers is far less rigid than it may appear from the above. Pipelined vector operations and parallel computations on distributed memory are two orthogonal architectural concepts to exploit parallelism. Distributed-memory parallel vector processors or superpipelined processors with large caches on a shared-memory bus are examples of hybrid combinations featured in commercial supercomputers available today. If an application proves to be portable from single vector processors to distributed-memory scalar parallel processors, one can safely assume that it will also achieve reasonable efficiency on a supercomputer of hybrid architecture. For the sake of clarity, we distinguish between single vector processors (which we call vector computers) and distributed-memory scalar parallel processors (which we call DMPPs), keeping in mind all variations in between.

The interest of the research community in DMPPs has led all leading vector supercomputer manufacturers to develop new supercomputers that are based on distributed-memory architectures. For many applications, however, DMPP operating systems and compilers are still incapable of assisting the user in obtaining a reasonable fraction of the machine's peak performance, in a way comparable with what can be expected on vector computers. Closing this gap faces many unsolved problems. Among others, the following three questions have yet to be answered.

- How difficult is the migration of existing vector supercomputer application codes to DMPPs, and by what techniques can parallelism expressed for vector operations be translated into parallel computations on data structures in distributed memories?
- How much can the system software (OS and compiler) ease the above migration, and how efficiently could a standardized software interface (for instance, a programming language like High Performance Fortran (HPF) [18]) support the implementation of scientific applications that are portable to both types of high performance architectures?
- How should large-scale applications be designed to achieve high performance on both vector and parallel supercomputers in a portable way?

It is easier to answer the questions above for applications requiring only highly structured computations, like dense matrix linear algebra, or finite element and difference methods applied to structured grids. In fact several research groups have defined Fortran-like languages and implemented compilers to allow portable coding of these applications. The definition of HPF is an attempt to standardize these efforts. The portability on a whole spectrum of architectures has been proven for some of these languages (see for instance [14]).

We are concerned with more unstructured computations, like linear algebra on general sparse matrices, or finite element or finite volume methods applied to nonuniformly refined grids. For this application class some software support must be provided at run-time; standard compile-time dependence analysis techniques fail when control-flow depends on run-time information. If we want to avoid "manual" intervention in the application code, run-time support should either be provided by the OS or by the compiler's run-time library. The three problems above have been discussed by only a few research groups [5], [21].

Strictly speaking, almost any unstructured solution method to a given scientific problem can be substituted by a structured method that is more easily parallelizable, but generally also more resource intensive. With many real-world problems, however, using a less powerful method induces a severe loss that cannot be offset even by massive parallelism. While there are many options to solve a particular problem, the only *efficient* solution method may require unstructured computations. Many publications reporting close to peak performance on some problem in fact use outdated regularly structured methods, and it would be faster to solve the same problem with a state-of-the-art unstructured method running at workstation performance.

This article tries to answer the three questions above for the existing vector supercomputer code PILS, a software Package of Iterative Linear Solvers targeted to the solution of very large, unstructured sparse, unsymmetric, ill-conditioned systems of linear equations. The package is used routinely in finite element-based applications on a range of sequential and vector supercomputers, and we want to extend this portability to DMPPs and among DMPPs. We have parallelized PILS using the compiler Oxygen. The performance of the generated parallel program is evaluated on two DMPP systems, namely the Intel Paragon [19] and the Fujitsu AP1000 [20].

A fully automatic parallelization of such a code would be impossible, just like today's highly mature vectorizing compilers are incapable of achieving high performance on a naive sequential implementation of such an application. Instead, carefully designed data structures together with additional hints to the compiler (usually in the form of directives) are necessary to exploit the architectural features of vector computers. We try to exploit the same restructuring effort in a highly vectorized application and substitute a parallelizing compiler for DMPPs, together with some additional tuning.

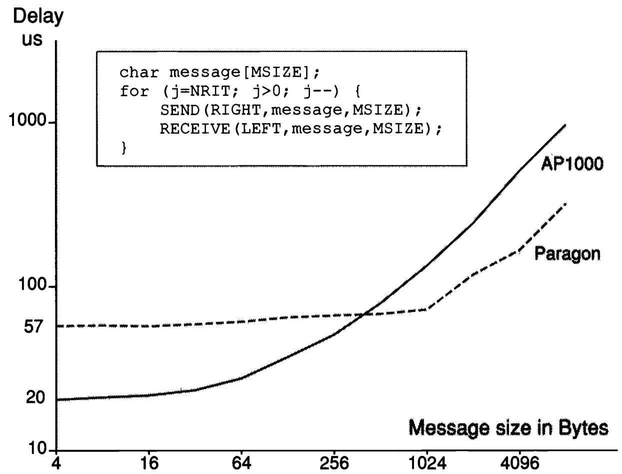
Our self-imposed restrictions include that modifications to the original application are kept to a minimum, so that the code stays fully compatible and compilable on vector computers. Except for the performance gain, the user should not be able to distinguish whether the application runs on a single processor or in parallel on the DMPP. Along with that, most communications are generated transparently by the parallelizing compiler, which hides the underlying message-passing architecture as far as possible.

Many recent studies investigate DMPP implementations of iterative solvers for linear systems. Almost all of these publications aim at a much simpler goal. Most authors consider only regularly structured problems, like 5-point finite difference discretizations of simple PDEs. The resulting matrices do not need a general sparsity data structure; all operations on them can be treated much more efficiently as addition, scaling, and positional shifts on smaller dense matrices. Furthermore, most authors do not consider preconditioning, except possibly for diagonal scaling. Good preconditioners often increase the convergence speed of an iterative method more drastically than parallelization, and many real-world ill-conditioned problems just cannot be solved by unpreconditioned iterative methods. On the other hand, good preconditioners, like approximate factorizations, are more difficult to parallelize than any other component of iterative solvers. Finally, many authors consider only symmetric positive-definite systems, for which conjugate gradients, but also weaker methods like successive over relaxation (SOR) or Jacobi iteration converge well. Despite significant new developments in the last few years, no single unsymmetric variant of conjugate gradients can be seen as universally successful on ill-conditioned unsymmetric linear systems, so that high flexibility in the choice of methods and preconditioners is required.

3. Background.

3.1. Target platforms. We have ported PILS to five different DMPPs: the CM-5 from Thinking Machines, the NEC Cenju-2 and Cenju-3, the Fujitsu AP1000, and the Intel Paragon. Results in this article are only drawn from the latter two.

3.1.1. Intel Paragon. The Intel Paragon XP/S 5+ features Intel i860 XP processors connected in a rectangular mesh. PEs operate at 50 MHz and contain 16 Kbyte data and instruction caches. The maximum bandwidth from cache to floating point unit is 800 Mbytes/s. Hardware communication bandwidth between any two nodes is 200 Mbytes/s full duplex. For this article, parallel measurements were taken on a machine with 32 Mbytes local PE memory,



```
char message[MSIZE];
for (j=NRIT; j>0; j--) {
    SEND(RIGHT,message,MSIZE);
    RECEIVE(LEFT,message,MSIZE);
}
```

FIG. 1. *Execution time of neighbor-to-neighbor communications in the (virtual) torus topology of* AP1000 *and Paragon for various message sizes. Both systems are configured as* 4 × 8 *tori. We use tagged communication operations as specified in the code segment above.*

whereas sequential measurements were performed on a single PE of another (much smaller) machine with 128 Mbytes.

As C compiler and back end to Oxygen, the PGI compiler version 4.5 was used under a beta release of operating system OSF AD 1.2.

The communication library of the Paragon is build on top of OSF. This provides a high functionality (like support of standard UNIX I/O library functions), but requires expensive context switches and thereby affects the utilization of the underlying communication hardware.

3.1.2. Fujitsu AP1000. An AP1000 consists of a collection of processors and a Sun-4 host. Each AP1000 PE consists of a 25 MHz Sparc with FPU, 16 Mbytes DRAM, and 128 Kbytes direct mapped cache memory. Three communication networks are available: the Torus network (T-net) for point-to-point communication between PEs, the Broadcast network (B-net) for host-to-PE communication, and the Synchronization network (S-net) for barrier synchronization. A detailed description of the AP1000 system can be found in [20].

For all our measurements we used the AP1000 OS CellOS1.1, with the Sun-C compiler SC1.0 as back end for Oxygen.

Each processor runs only a relatively primitive microkernel that supports communication to the host and to other processors. The host connects to the outside world and provides general I/O.

3.1.3. Basic measurements. We showed in [2], [3] that the speedup of DMPP programs strongly depends on the DMPP's *communication/computation ratio.*

To determine communication performance, we measure execution time of neighbor-to-neighbor communications in the (virtual) torus topology for various message sizes. Figure 1 compares such measurements on Paragon and AP1000. Both machines are configured as 4 × 8 tori.

The performance of floating point operations is difficult to predict on complex computer architectures, as it depends strongly on the location of the operands in the memory hierarchy. We have therefore measured T_c, the time required to compute a double-precision multiply add operation as in DAXPY, for in-cache and out-of-cache operands. Table 1 shows the results

TABLE 1

Measurements of computation speed (in C-compiled DAXPYs) on the two systems. For comparisons we also include measurements for several Sparc-based workstations, namely Sun SparcStations 1+, 2, *and* 10–41.

	T_c (μs)				
	Paragon	AP1000	SS1+	SS2	SS10
In cache	0.27	1.15	1.93	0.87	0.24
Out of cache	1.14	1.91	2.25	1.18	0.58

for the two DMPPs. Note that code generated by the Paragon C compiler even for the simple DAXPY loop does not run close to peak performance.

3.2. PILS. PILS implements a large number of iterative methods, preconditioners, and other variants for iterative solvers and provides a high degree of flexibility, like automatic adaptation to a more robust preconditioned iterative method whenever the solver that is usually fastest fails to converge. The library is integrated into a number of applications, including several semiconductor device simulators, where the solution of hundreds of ill-conditioned linear systems of similar sparsity structure dominates the overall execution time. PILS and its client applications have been used regularly over four years now, at several dozens of academic and industrial sites.

PILS runs on a number of different platforms, from workstations to supercomputers, including Convex, Cray, Fujitsu, and NEC vector computers. The sparse matrix data structures in PILS are optimized for the use of vector computers.

This section describes those features of PILS that are relevant for our project. Interested readers are referred to [29] for an overview of algorithms and applications, [30] for a performance analysis, and [27] for ample detail on algorithms and implementation.

3.2.1. Colored jagged diagonals based on a partitioning by matchings. Nonzero entries of the sparse matrix are stored in so-called "jagged diagonals" [33], a data structure in which matrix–vector products can be computed with a few long vector operations, even if the number of nonzeros per row or column is small. Every operation on a row-oriented jagged diagonal (where row indices are given implicitly by the storage order, and column indices are stored explicitly in an index vector) accesses one of the argument vectors (on the right-hand side of an assignment) through indirect addressing, using a vector-gather routine.

Column-oriented jagged diagonal operations update a vector by addressing it indirectly both on the left- and the right-hand side of an assignment, requiring vector-gather and vector-scatter routines. To avoid write conflicts in this scatter, the entries in the index vector must be unique. This is obtained through a reordering of the nonzeros resulting from a partitioning by matchings in the associated graph. PILS uses a combination of row- and column-oriented jagged diagonals to save storage on structurally symmetric parts of the matrix and to make transposed matrix–vector multiplication as efficient as matrix–vector multiplication without transposition. Transposed matrix–vector products are important in some of the iterative methods, and are inefficient with many other sparse matrix formats.

The rows of the sparse matrix are further symmetrically permuted into blocks, such that the blocks comprising the diagonal are zero everywhere except on the diagonal itself. This allows for a vectorized solution of sparse triangular systems, which is important for fast no-fill incomplete factorization preconditioners. Such a block structure is obtained after coloring the associated graph of the matrix into sets of independent vertices. Every color block of each triangle of the sparse matrix is stored in separate jagged diagonals.

3.2.2. Language usage for portable vectorization. To meet the requirement for the data structuring and flexibility, PILS is mainly implemented in C++. Because of the existence of a translator to C [37], C++ is available on all machines with a C compiler, including all vector supercomputers. Unfortunately, the code generated by vectorizing C compilers, if available at all, is generally slower than code generated by vectorizing Fortran compilers. For this reason, the most time-critical, vectorizable parts of PILS are written in Fortran. These Fortran parts consist of 46 simple loops and four nested loops. Although they account for only six percent of the total code size, these loops take over 99 percent of the solver's run-time in a typical application and are fully vectorized. The compilation system inserts for each machine the appropriate compiler directives, allowing the vectorizing compiler associative rearrangements of global summations, or asserts that index vectors used on the left-hand side of an assignment are unique and safe for scatter operations.

In particular, all linear algebra operations are handled by these critical loops. Addition, scaling, componentwise multiplication, copying, and zeroing of vectors as well as inner products are among the simple Fortran loops.[1] Sparse matrix operations are supported through several more complicated loops that use indexed addressing and deeper nesting.

The C++ part of PILS constructs data structures, interprets specifications to set up the correctly parametrized solution method, and interfaces with the outside world (e.g., a client application), converting external formats into internal data structures if required. Within the solution process in a preconditioned iterative method, C++ controls the program flow, calls linear algebra routines, and checks convergence through vector norms, but never accesses directly the entries of any of the vectors involved in the iteration. In the computation of sparse matrix–vector products, as well as in the application of no-fill incomplete factorization preconditioners, a C++ program segment guides through the data structure, but the final computation involving the argument and result vectors is performed by Fortran segments.

3.2.3. PILS client applications. Typical client applications for PILS require the solution of many linear systems in which the matrices have different numerical values but only one or a few sparsity structures. These matrices arise, e.g., in the discretization of nonlinear PDEs into a system of nonlinear equations. Every iteration of a Newton solver for this nonlinear system requires the solution of a linear system of equations. Similarly, the solution of time-dependent PDEs calls for the solution of one or more linear systems at each time step. Semiconductor device simulation, our target application [16], requires the solution of up to several hundreds or even thousands of linear systems within the modeling of a system of time-dependent nonlinear PDEs on nonuniformly refined two- or three-dimensional finite element grids. For large simulations, the time to solve linear systems accounts for more than 95 percent of the overall execution time.

As the sparsity structure of the matrices is reused many times, the initialization cost for data structures is amortized over several numerical solver runs. The time used in the inherently sequential algorithms to build vectorizable data structures is negligible in comparison to their benefits.

The linear systems occurring in device simulation can be very ill conditioned. Occasionally, iterative methods preconditioned by standard fast no-fill incomplete factorization preconditioners fail to converge. In such a case, PILS adapts to more robust preconditioners, like approximate factorizations with numerical dropping of fill entries based on a threshold tolerance. Unfortunately, these robust preconditioners offer little room for vectorization or parallelism and run essentially in scalar mode.

3.3. Oxygen. Oxygen is a parallelizing Fortran compiler for DMPPs. It accepts Fortran 77 and compiler directives and generates parallel C code with communication primitives

[1] Note that the collection does not consist only of BLAS1 routines [23]. Most critical loops perform combined operations that allow a more efficient usage of caches, vector registers, and chainable pipelines.

for execution on DMPPs. Originally designed as part of the K2 project [1], the compiler has been ported to several platforms. For a list of such platforms and for an evaluation of the efficiency of Oxygen-generated parallel programs, the interested reader is referred to [31].

Several compilation systems with similar (or more) functionality than Oxygen have been developed recently [40], [24], [17], [22], [13], [41], [6]. Such compilers provide programming comfort and ease program portability on DMPPs and often target automatic parallelization of existing "dusty deck" sequential Fortran programs. As a result of those efforts the HPF standard has evolved [18], which essentially defines a set of Fortran annotations to distribute data and parallelize loops on DMPPs. In all the above systems, distributed data can be accessed through what is commonly referred to as a *global name space*. This means that any processor's reference to an element of a distributed data structure is handled by the compiler, transparently to the user; the compiler inserts communication primitives in the generated DMPP program to allow such remote data references.

For this study, Oxygen serves only as a tool to ease the parallelization of PILS, without making use of its automatic parallelization capabilities. While simple programs like dense linear algebra codes such as LINPACK can be parallelized automatically in an efficient way [26], it is commonly believed that strong user interaction is required to parallelize PILS-like codes, even on simpler parallel architectures such as shared-memory multiprocessors.

The Oxygen input language features a global name space much like HPF. In contrast to the above-mentioned compilers of languages that had some influence on the HPF definition, and also in contrast to the first HPF compilers commercially available (e.g., the HPF subset compiler from Applied Parallel Research [4]), Oxygen includes several features that make it especially well suited for supporting the parallelization of unstructured computations in general and PILS in particular:

- Oxygen directives include constructs to dynamically distribute data and control flow in parallel programs.
- Run-time analysis supports a global name space even in program segments that include arbitrarily nested dependencies on elements of distributed arrays.
- Not only remote fetches, but also *remote updates* of distributed data, are allowed. That is, we do not restrict ourselves to the "owner-computes rule."
- All processors in Oxygen-generated parallel programs execute duplicates of the sequential part and synchronize only implicitly through local communications as required by data exchanges.

Some of the above features are supported also by few other experimental systems (for instance, remote updates by ARF [40]). However, for the DMPP parallelization of PILS, *all* these features are crucial.

3.3.1. Programming paradigm. Most of the systems mentioned above support a global name space primarily through compile-time analysis. The user can specify data distributions in the input language, and the compiler then distributes operations on distributed data by using, for instance, the owner-computes rule; i.e., only the PE that allocates a data item is allowed to change it. To a limited extent, some of the above systems (for instance Fortran D and ARF) support the global name space also through run-time analysis. For code segments for which the compiler cannot statically determine from where to fetch nonlocal data (for instance, when sparse matrix data structures are used), the compiler generates two execution phases: a so-called *inspector* and an *executor*. The inspector preprocesses execution to find out about nonlocal data accesses. The executor performs the actual computation and uses communication patterns constructed by the inspector to fetch nonlocal data. Such run-time analysis introduces significant overhead in execution time. However, many important applications that require such run-time analysis have two important characteristics: (1) the critical code segments are invoked repeatedly, for instance as computational kernels of an iterative solution method; (2) the critical code segments can be *start-time scheduled*; i.e., although control flow depends on run-time information, once a communication pattern has been computed, it can be reused for repeated execution of a given segment.

Oxygen supports a global name space primarily at run-time. It also generates two execution phases for critical code segments: *symbol handlers* and executors. We do not denote the preprocessing phase as inspector because the symbol handler has more functionality: while inspectors can only handle a single level of indirect array accesses, the symbol handler correctly preprocesses code segments with multiple nesting levels of dependencies on nonlocal data. The compiler generates several preprocessor *slices* [10], i.e., symbol-handler iterations that generate communication patterns either for the following symbol-handler iteration or for the executor. Oxygen's run-time analysis is also more powerful in saving communication patterns: driven by compiler directives, for a given code segment, several communication patterns can be saved; communication patterns are stored symbolically, such that preprocessing can be avoided even when a computational kernel is applied to different data (for instance by calling a subroutine with different parameters), as long as the data structures have identical shape. Note that the generation of recursive inspectors has recently also been investigated by other research groups [6], [7].

Oxygen lets the user decide where computation on distributed data is carried out, and both remote fetches and updates of nonlocal data are allowed. Therefore, the input language not only includes directives for data distribution but also for loop distribution. Both data and loops can be distributed dynamically using a user-defined mapping computed at run-time. Unless specified differently by the user, variables are replicated (and not part of the global name space), and sequential code is executed on all processors. Many of the above other systems imply global synchronizations at the end of parallel loops by restricting execution of sequential program parts to one processor or by using collective communication statements to orchestrate nonlocal data accesses. Oxygen generates parallel programs where PEs synchronize only implicitly through the use of local communication primitives (i.e., nonblocking sends and blocking receives) for nonlocal data exchange.

4. Porting PILS with Oxygen. If the linear solver is the main time-consuming operation within an application implemented using PILS, we can obtain a notable benefit by parallelizing only PILS, without touching the client application. Parts with significant parallelism are already marked in PILS as vectorizable parts. These critical loops, mentioned in §3.2.2, are parallelized by Oxygen.

The client application should preferably run sequentially and not see a difference between a sequential or a parallel version of the PILS library. Because of the large memory requirements of a typical client application, and because of necessary OS support, we decided on the AP1000 to run the client application and the mostly sequential parts of PILS that construct data structures on the AP1000 host. On the Paragon, any PE can take the role of the host, since virtual memory and standard UNIX I/O library routine calls are also supported on compute nodes. We decided that Paragon PE 0 both executes the sequential control code and its share of the parallelized code. An alternative model, where one particular PE executes *only* sequential code, was discarded because—for increased parallel efficiency—Oxygen requires the underlying machine to feature a number of processors computable as a power of two. To simplify the following discussion we call Paragon PE 0 as well as the AP1000 host *controller*.

The separation of the numerically intensive, time-consuming part of PILS coded in Fortran and the more complex control and interfacing part coded in C++ led us to the following parallelization strategy:

- All C++ code is executed on the controller.
- All Fortran routines are annotated with parallelization directives and compiled by Oxygen into parallel subroutines to be executed on the PEs.
- Controller and PEs communicate via a remote procedure call interface. This interface requires an additional software layer between C++ and Fortran that determines which parameters to send from controller to PEs on subroutine invocation, and which parameters to return at the end of the routine's parallel execution.

We will now describe the parallelization of the numerically intensive time-consuming Fortran source on the PEs and then explain how the remote procedure call interface was optimized to avoid expensive controller/PE communications.

4.1. Parallelizing the Fortran part. The numerically intensive part of PILS consists of linear algebra operations on vectors and sparse matrices. As mentioned in §3.2.1, matrices are stored in colored jagged diagonals. The part of this data structure that describes the sparsity structure consists mainly of indices for nonzero entries and remains constant while an application is solving many linear systems with the same sparsity structure. The numerical part of the data structure, with one particular instance of values for the nonzero entries, remains constant over the whole iterative algorithm that solves one particular system of linear equations. The average number of nonzeros per row in the matrices ranges typically between five and twenty-two, and there are typically between five and twenty vectors involved in the iteration process. Both matrices and vectors thus have to be distributed for memory and execution-time efficiency.

Matrices can be distributed in a way that they are read only locally during the solution of one linear system. Distributed vectors are accessed remotely and updated in each iteration. We declare them as distributed Fortran arrays, so that remote accesses to vector elements are supported by Oxygen's global name space.

All vectors share the same dynamic distribution in all fifty subroutines. A distribution vector specifying ownership of vector elements (i.e., which element is allocated on which PE) is defined in data distribution directives when vectors are declared. The distribution vector is initialized by the controller/PE interface when the solver starts executing.

The same dynamic partitioning and mapping is also used for simple loops. This makes efficient parallelization of many Fortran routines defining vector–vector operations rather simple, as no linear operation on vectors requires communication. Since Oxygen does not enforce global synchronization at the end of loops, PEs compute fully asynchronously on these operations. Only a few such routines (for instance, the computation of a dot product) require global reduction operations.

The indirect addressing within sparse matrix–vector products constitutes the major challenge. The same operations that require vector-gather routines on a vector computer (see §3.2.1) need remote memory fetching on a DMPP. Oxygen handles these transparently by generating appropriate messages for accesses to the global name space. Since the indirect addresses are not known at compile-time, the communication patterns can only be generated at run-time. As the same patterns are required many times (there are one or two matrix–vector products per iteration, and several tens of iterations to solve a linear system), the executor can reuse communication patterns, and the time-consuming symbol handler is performed only once.

Vector-computer scatter routines writing to an indexed array correspond to remote updates on a DMPP. These operations can be parallelized efficiently only because Oxygen does *not* restrict computation to the owner of a data item. The generation of remote update messages is just as transparent and efficient as remote fetching.

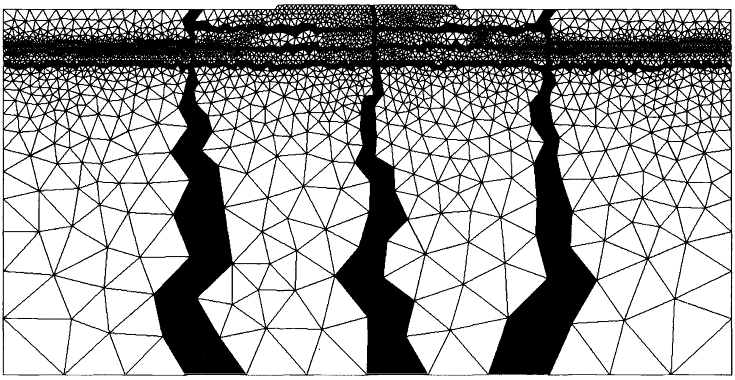


FIG. 2. *Two-dimensional geometric mapping of a refined discretization grid to a* 4 × 4 *processor interconnection network. All elements* (*triangles*) *whose corners are assigned to different processor are drawn in black. White lines crossing black strips correspond to graph edges that require communication; local edges are drawn as black lines.*

4.2. Mapping for locality. Any simple-minded assignment of vector entries to processors would degrade performance disastrously, as almost all indirectly addressed entries would result in remote accesses, and every processor would have to communicate with every other processor. The *mapping problem* for finite element meshes consists of partitioning vectors (and matrix rows and columns) in a way that each processor owns an approximately equal number of entries (for load balancing) and a maximum number of accesses are local. Moreover, a minimum number of other processors should be accessed for remote fetching or updating, and these processors should be at minimum distance in the communication network (thus reducing transfer latency and network contention). The development of heuristics for this (NP-hard) mapping problem is currently a field of active research, and beyond the scope of a parallelizing compiler, but the possibility of defining a dynamic mapping using a distribution list allows us to add problem-specific mapping heuristics to our code.

We selected a two-dimensional geometric mapping heuristic that we had good experience with in an earlier related project [28] and that is well adjusted to the two-dimensional network topologies of both the Paragon and the AP1000. The heuristic proceeds as follows on n vertices mapped to a $p_v \times p_h$ torus or mesh of processors: First, vertices are sorted according to their x-coordinate in the physical discretization grid. This sorted list of vertices is partitioned by $(p_v - 1)$ vertical cut lines into p_v sets of n/p_v vertices each, and each of these sets is assigned to one column of the mesh of processors. Then each set is sorted again according to the y-coordinate and partitioned by $(p_h - 1)$ horizontal cut lines into p_h sets, and each of these sets is assigned to one processor of this column. This algorithm has a time complexity of $O(n \log n)$ and can be used in the sequential part on the controller, as it does not dominate over other algorithms in the construction of the colored jagged diagonal data structure. Figure 2 shows how the discretization grid of the smallest of our test cases was mapped on 16 processors arranged in a 4 × 4 mesh.

Recently, new mapping heuristics such as recursive spectral bisection [35] have been shown to reduce the total number of edges that are cut by processor boundaries. We refrain from such techniques because the eigenvalue computations required in these algorithms would clearly dominate over all other operations in the construction of the PILS data structures. The normal procedure would be to move these computations off line to a separate mapping program whose result is read from the file before starting the parallel program. This would conflict too much with our goal to parallelize PILS transparently from its client application.

4.3. Coloring. Incomplete factorization preconditioners rely on the solution of sparse triangular systems of linear equations. In the case of fast preconditioners with no fill (like ILU or its faster variant D-ILU), the sparsity structure of the triangular matrices for the preconditioner coincides with the corresponding triangle of the system matrix.

The solution of triangular systems imposes higher sequentiality than matrix–vector multiplication. In a lower triangular system $Lx = b$, for instance, the ith entry of the solution vector x_i depends on the calculation of the previous entries, x_j with $j < i$. Apparently, this would impose a synchronization point (with data exchange through communication) for each row. However, sparsity in L reduces the dependencies: x_i depends only on x_j if ℓ_{ij} is nonzero. *Coloring* is a technique to exploit this sparsity in order to construct higher parallelism in triangular system solvers.

Coloring is already used in PILS to vectorize triangular system solvers (see §3.2.1) and is integrated into the colored jagged diagonal data structure. The solution of a triangular system with c colors now results in $(c - 1)$ self-synchronizing data exchanges, adding up to $(2c - 2)$ communication points for a D-ILU preconditioned matrix–vector product. The typical number of colors is around 7 for our two-dimensional grids and around 12 for our three-dimensional grids.

Even if perfect load balance in vector–vector operations and unpreconditioned matrix–vector multiplication is achieved by mapping an equal number of vector entries to each processor, the coloring introduces some imbalance within the processing of each single color. The *balanced greedy coloring heuristic* [28] achieves an equal distribution of each color, at the expense of increasing the total number of colors. As this type of coloring would require certain modifications in the code that handles the jagged diagonal sparse matrix data structure, we refrained from this additional tuning.

4.4. The controller/PE interface. The controller/PE interface replaces Fortran subroutine calls in the C++ PILS source code with a remote procedure call (RPC) interface. It first passes all parameters required for the parallel execution of the subroutine to the PEs, then initiates the start of the corresponding compiler-parallelized routine on the PEs, and finally, after the parallelized routine's execution, the interface collects results necessary on the controller for further execution of the solver. Parameters to be passed include scalars, vectors, and sparse matrices.

For best performance, matrix assembly should be performed in parallel on the PEs. However, as most client applications assemble their matrices outside of PILS, we provide a mechanism to load the matrix from the controller to the PEs, so that the entire client application can run on the controller without necessarily having to be parallelized as well. Correct parallel execution does not require the matrix to be loaded for each parallel matrix–vector operation. As already mentioned above, the matrix is only read accessed and can therefore be loaded only once for each linear system.

Controller-to-PE distribution of vectors jeopardizes efficient parallel execution of any vector operation, if read access to a vector during such operation requires the controller to distribute the vector to the PEs. Iterative solvers spend most of their time updating vectors. Because the PILS source was optimized such that the most time-consuming code was implemented in Fortran, most vector accesses take place *in the Fortran code*; i.e., the controller never overwrites a vector during the iterative solver's execution. Most dependencies from the Fortran subroutines to the controlling C++ code are due to data items involved in control flow decisions, such as the computation of a convergence criterion.

The above observation allows for heavy optimization of the controller-to-PE communication protocol: if indeed a vector is only modified by the Fortran subroutines, it must be distributed to the PEs only once; any future subroutine accessing the same vector will operate on the same data kept resident in the PE memory after the subroutines finish execution.

However, without major changes in the controlling PILS C++ source, it is difficult to decide statically whether any valid vector copy exists on the PEs or whether such a vector should be (re)initialized by the controller/PE interface. Our measurements in §5 show that such a decision can be made efficiently at run-time by the controller/PE interface. As part of that interface we have implemented on the PEs a *controller/PE cache*, which dynamically allocates and downloads from the controller copies of vectors accessed in parallelized Fortran subroutines. Whenever the cache on the PEs finds a valid memory-resident copy of a vector it does not download the vector from the controller.

For result vectors and scalars that are used in the controller code after they are modified by a parallelized Fortran subroutine, the PEs are forced to invalidate the corresponding entry in the controller/PE cache. Such a decision is done statically as part of the controller/PE interface code.

On some platforms controller/PE communications are rather expensive. On the AP1000, for instance, PEs are attached to the host via a VME interface, which is much slower than interprocessor communication channels.

The following two strategies were chosen to further optimize the interface. First, only few parallelized Fortran subroutines actually return data to the controller. These are mostly vector dot products used in the iterative algorithm to compute scalars and control termination. Therefore, the execution of most subroutines can be *pipelined*, which reduces PE idle times due to controller-to-PE latency *and* PE idle times due to interprocessor communication. Oxygen does not introduce global synchronizations at the end of most parallel loops. Second, the controller/PE cache is also implemented on the controller to reduce protocol overhead between controller and PEs. The controller itself computes which data need to be distributed; the PEs take in parallel the same decision about which data need to be received from the controller.

4.5. Visualization. To summarize our porting strategy, we visualize in Fig. 3 different execution phases of a PILS application running in parallel on a 2×4 Paragon system. For the visualization, PILS was instrumented to drive the performance analysis tool ParaGraph [15], which is part of the Paragon programming environment. The following different phases of execution are shown together in a "Task Gantt Chart" [12]: sequential execution of C++ code on PE 0, parallel (and asynchronous) execution of Fortran routines, and the run-time analysis introduced by Oxygen during the first PILS iteration.

The symbol-handler overhead only appears in the first iteration. Later iterations save execution time by reusing communication patterns. Further overhead in the first iteration on PE 0 is due to sequential operations like preconditioner setup. The asynchronous overlapping of computation in different processors is clearly visible during colored preconditioned matrix–vector multiplications using gather and scatter remote accesses.

5. Experimental results.

5.1. Test environment. Our benchmark suite consists of six linear systems extracted from real semiconductor device simulations. These problems are selected to display the variety of problem sizes and complexities within typical large device simulations. Most of our problems are based on three-dimensional discretizations, as two-dimensional device simulation problems are often small enough to be solved more efficiently with sparse direct solvers.

Standard device simulation solves the drift-diffusion equations, a set of three PDEs in three variables. Depending on the operating mode, the discretized equations are solved separately

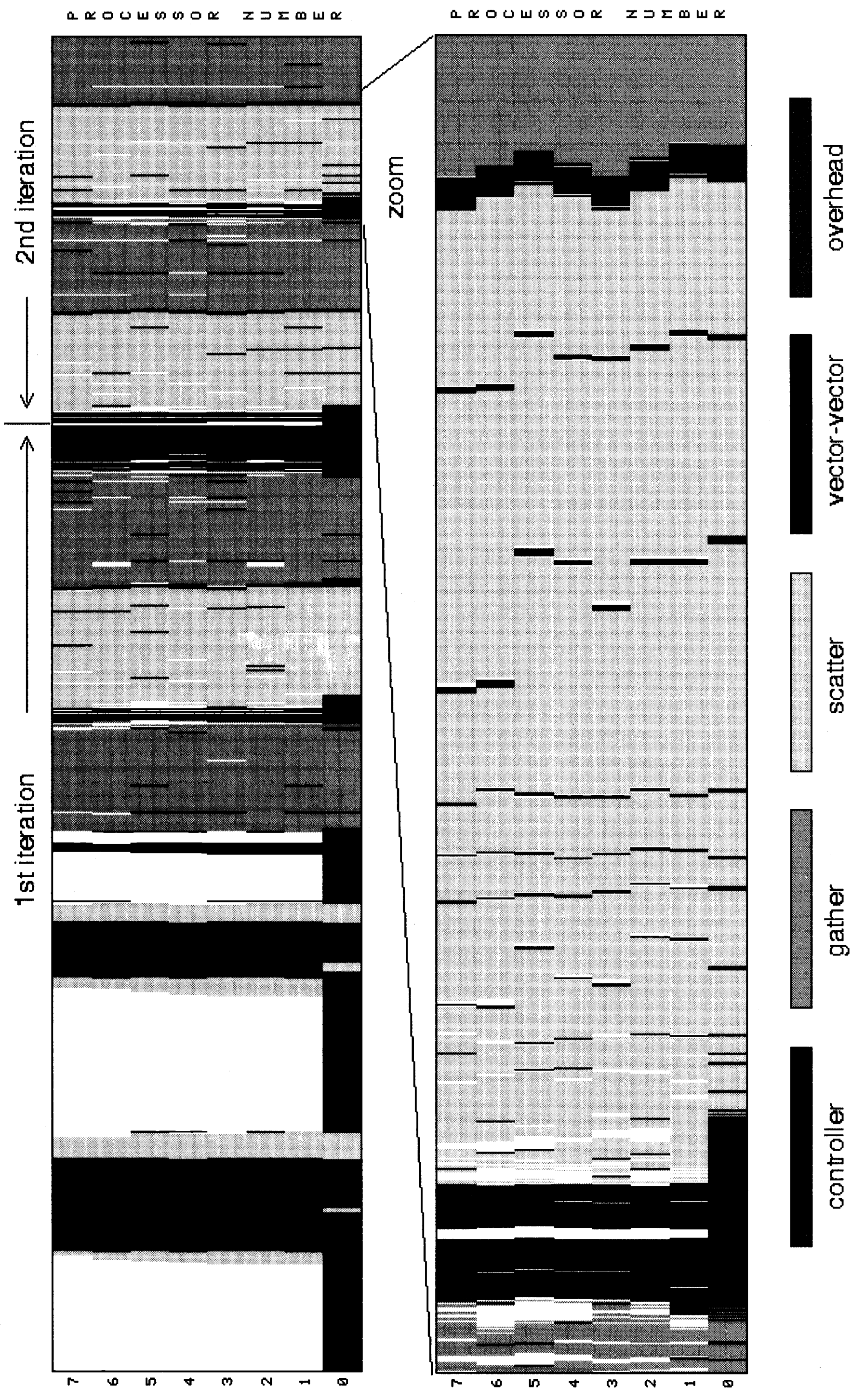


FIG. 3. *The parallel execution of PILS is visualized using Task Gantt charts of ParaGraph. Different execution phases of PILS running on a* 2 × 4 *Paragon are shown in different shades of gray. We distinguish controller execution, Oxygen symbol-handler overhead, and actual parallel computation of different PILS library routines. The top Gantt chart depicts the first and most of the second Bi-CGSTAB iteration; the bottom chart zooms into the solution of a sparse triangular system within a preconditioned matrix–vector multiplication.*

TABLE 2
Problem characteristics.

	LDDH	DR15E	BIPOl3D20KH	BP25E	DR15C	BP25C
Grid dimension	2-D	3-D	3-D	3-D	3-D	3-D
# PDEs	1	1	1	1	3	3
# unknowns	2674	15564	20412	25642	46692	76926
# nonzeros	18614	143710	263920	234436	986042	1618414
Matrix density	7.0	9.2	13.0	9.1	21.1	21.0

("Gummel iteration"), involving one unknown per discretization grid point, or all three PDEs are assembled in a coupled system with three unknowns per grid point. Grid dimensionality and number of PDEs influence the density of the sparse matrix and the connectivity for partitionings as those used in the mapping. Table 2 summarizes the characteristics of our test problems. Matrix density is conveniently expressed as the number of nonzero entries per row. Unless otherwise noted, all benchmarks are done using Bi-CGSTAB as the iterative solution method, preconditioned by a D-ILU preconditioner in split position.

5.2. Parallel reduction operations and convergence. Iterative solvers are sensitive to rounding errors in the computation of vector dot products. Any parallel implementation of such a reduction operation must modify the summation order with respect to the sequential operation. Since floating-point addition is not associative, even a slight change in the summation order induces a different rounding of the result. In an iterative solver, these rounding variations propagate up to variations in the total number of iterations required to solve a given problem (for very sensitive ill-conditioned problems, such variations may even result in divergence or algorithmic breakdown [27]).

Vector dot products are executed in parallel on a DMPP by summing up globally local dot products on the local partial vectors. Oxygen substitutes tuned library calls for these global summations. Unfortunately, the summation order in these manufacturer-provided library routines typically depends on the number of processors of the underlying machine configuration.

In Table 3 we have collected the number of iterations required for both serial and parallel solver runs. Due to the different sequences of summations when computing vector dot products, the required number of iterations to achieve a given precision (here 10^{-10}) with the parallelized solver may be both smaller or larger than for the serial program run. In order to avoid misinterpretations due to this effect, whenever the measurements presented in this section should demonstrate the efficiency of our parallelization strategy, we refer to only one iteration of the iterative solver. Total execution times are shown only when different solution methods are compared.

5.3. Execution time and speedups per iteration. The total execution time of PILS can roughly be decomposed into three components: (1) execution time of the compute intensive Fortran code, (2) distribution of the matrix from controller to PEs, and (3) overhead of the controller/PE RPC interface. The first component is depicted in Table 4; measurements of the second and third components are summarized in Table 5.

Table 4 summarizes execution times and speedups measured for one iteration running on one Paragon and one AP1000 processor, respectively, on both a SparcStation 1+ and a SparcStation 2, and running in parallel on Paragon and AP1000 systems of different sizes. The parallel execution times refer to the Fortran code only; i.e., no controller/PE communication overhead is included. On the one hand, parallel performance depends on the structure of the problem: larger systems like DR15C achieve high speedups, because the relative importance of the communication overhead decreases. On the other hand, different communication

TABLE 3
Number of iterations required to achieve precision 10^{-10}.

	LDDH	DR15E	BIPOL3D20KH	BP25E	DR15C	BP25C
Serial execution						
	56	71	80	30	65	102
Parallel execution						
Paragon 4 × 4	59	71	80	30	65	90
Paragon 4 × 8	56	71	77	30	65	90
Paragon 8 × 8	58	71	81	30	65	94
AP1000 4 × 4	55	71	69	30	65	n.a.
AP1000 4 × 8	58	71	82	30	65	n.a.
AP1000 8 × 8	54	71	80	30	65	89
AP1000 8 × 16	54	71	80	30	65	96

TABLE 4
Execution times per iteration in seconds and speedup of the PILS Fortran code. Table entries denoted with n.a. (not available) could not be filled due to the memory requirements of the largest problems. Numbers in italics have been extrapolated using SparcStation measurements.

	LDDH	DR15E	BIPOL3D20KH	BP25E	DR15C	BP25C
Serial execution times (s)						
Paragon	0.066	0.37	0.60	0.60	2.11	3.43
AP1000	0.20	1.46	*2.4*	*2.5*	*8.6*	*14.2*
SparcStation 1+	0.16	1.15	1.89	1.98	6.72	n.a.
SparcStation 2	0.09	0.69	1.13	1.15	4.03	6.68
Parallel execution times (s)						
Paragon 4 × 4	0.041	0.105	0.266	0.249	0.616	1.632
Paragon 4 × 8	0.043	0.079	0.177	0.159	0.368	1.093
Paragon 8 × 8	0.040	0.076	0.133	0.138	0.285	0.579
AP1000 4 × 4	0.035	0.188	0.326	0.474	1.059	n.a.
AP1000 4 × 8	0.025	0.104	0.182	0.295	0.571	n.a.
AP1000 8 × 8	0.019	0.067	0.109	0.176	0.346	0.750
AP1000 8 × 16	0.017	0.045	0.072	0.127	0.213	0.500
Speedup						
Paragon 4 × 4	1.6	3.5	2.3	2.4	3.4	2.1
Paragon 4 × 8	1.6	4.7	3.4	3.8	5.7	3.1
Paragon 8 × 8	1.7	4.9	4.5	4.4	7.4	5.9
AP1000 4 × 4	5.7	7.8	*7.4*	*5.3*	*8.1*	n.a.
AP1000 4 × 8	8.0	14.0	*13.2*	*8.5*	*15.1*	n.a.
AP1000 8 × 8	10.5	21.8	*22.0*	*14.2*	*24.9*	*18.9*
AP1000 8 × 16	11.7	32.4	*33.3*	*19.7*	*40.4*	*28.4*

performance of the underlying DMPP strongly influences speedup measurements. As also shown in Fig. 1 and Table 1, for short messages, the ratio of communication to computation performance is approximately eight times smaller on the AP1000 than on the Paragon. It is therefore not surprising that the highest speedups could be measured on the AP1000. The lower message latency also lets the AP1000 outperform a Paragon with the same number of processors on moderately sized problems.

The major reason for the measured differences on the Paragon and the AP1000 is the different communication performance of the two systems. In order to quantify how important low communication startup latencies are for the efficient distributed-memory parallel execution of PILS, we have depicted in Fig. 4 the total number of PE-to-PE messages communicated in an 8 × 8 AP1000 as a function of the message size for the complete solution of each

TABLE 5
Average execution time per iteration (measured on the controller) and total overhead for the matrix distribution; the distribution overhead is given both as time (in seconds) and as controller/PE bandwidth (in Mbytes/s).

	LDDH	DR15E	BIPOL3D20KH	BP25E	DR15C	BP25C
Controller execution time (s)						
Paragon 4 × 4	0.058	0.124	0.288	0.264	0.644	1.672
Paragon 4 × 8	0.059	0.098	0.198	0.175	0.398	1.115
Paragon 8 × 8	0.056	0.095	0.155	0.155	0.316	0.604
AP1000 4 × 4	0.103	0.258	0.402	0.523	1.160	n.a.
AP1000 4 × 8	0.100	0.174	0.259	0.349	0.672	n.a.
AP1000 8 × 8	0.094	0.139	0.187	0.237	0.442	0.844
AP1000 8 × 16	0.094	0.117	0.151	0.187	0.310	0.589
Time for matrix distribution (s)						
Paragon 8 × 8	0.162	0.690	1.138	0.795	5.155	5.839
AP1000 8 × 8	0.557	3.347	5.937	4.270	21.957	28.903
Controller/PE bandwidth for matrix distribution (Mbytes/s)						
Paragon 8 × 8	0.393	0.742	0.856	1.051	0.729	1.056
AP1000 8 × 8	0.113	0.152	0.164	0.195	0.171	0.213

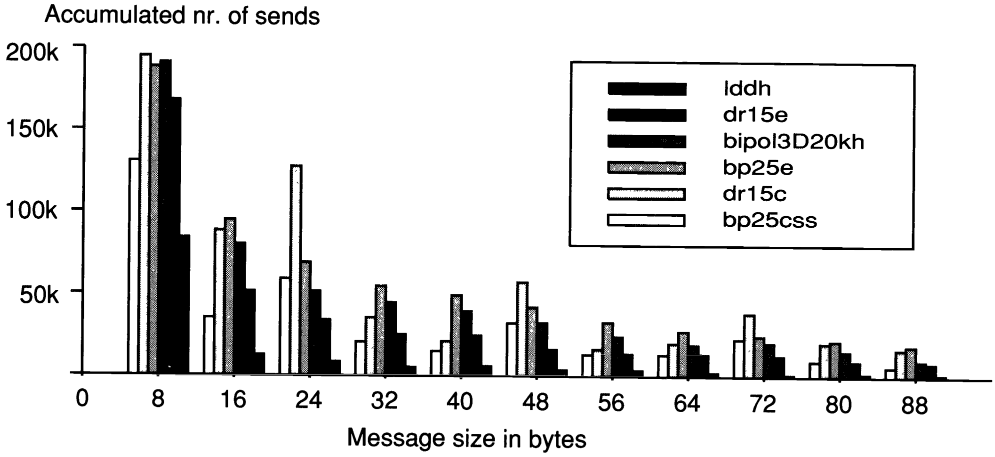


FIG. 4. *Distribution of message sizes for parallel execution on an* 8 × 8 AP1000 *system.*

of the six problems. From this figure it is obvious that very often only small messages are communicated. Since the compiler is able to pack messages optimally at run-time, this distribution only depends on the partitioning strategy and the resulting data dependencies.

5.4. Introducing global synchronizations. Most of the compilation systems mentioned in §3.3 generate collective communication statements at the end of each parallel loop. Therefore at any such point an implicit synchronization of all PEs takes place. When an Oxygen-generated parallel program is executed, only PEs that need to exchange data implicitly synchronize by blocking-receive statements.

To demonstrate what impact global synchronizations have on the overall performance in Table 6 we compare PE execution times per iteration on one hand running an example problem (DR15C) without synchronizations on several Paragon configurations, and on the other hand running the same code with an additional synchronization at the end of each parallel loop. Performance degrades quickly with increasing number of processors. This is partially because

TABLE 6

PE execution times per. iteration (in seconds) without and with global synchronizations introduced after every parallel Fortran loop of the program running on several Paragon configurations.

Paragon configuration	Without synchronization	With synchronization
2×2	1.799	2.269
2×4	1.022	1.765
4×4	0.616	1.630
4×8	0.368	1.626
8×8	0.285	1.830

TABLE 7

Execution time (in seconds) of various variants of methods for one particular problem on 64 processors of an Intel Paragon and a Fujitsu AP1000. *The numbers in parentheses indicate speedup over a single processor. This problem is too large to fit on a single* AP1000 *processor, so we extrapolated the* AP1000 *speedup from SparcStation measurements.*

Method	Preconditioner	Iterations	Time (Speedup)	
			Paragon	AP1000
Reference				
Bi-CGSTAB	split D-ILU	65	18.5 (7.4)	23.1 (24.9)
Other methods				
BiCG	split D-ILU	130	40.4 (6.4)	45.9 (25.6)
CGS	split D-ILU	78	21.6 (7.4)	27.2 (26.2)
GMRES(10)	split D-ILU	388	67.5 (7.6)	87.4 (25.4)
GMRES(∞)	split D-ILU	89	37.9 (7.0)	41.6 (31.6)
Other preconditioners				
Bi-CGSTAB	right D-ILU	67	28.7 (6.6)	33.4 (23.5)
Bi-CGSTAB	split ILU	61	33.1 (6.9)	38.0 (23.8)
Bi-CGSTAB	split SSOR	69	19.5 (7.3)	24.6 (23.6)
Nested iterative solvers				
GCR(∞)	GMRES / D-ILU	17	35.7 (7.0)	43.0 (25.3)
GCR(∞)	Bi-CGSTAB / D-ILU	11	35.1 (6.5)	41.4 (23.3)

synchronizations *per se* cost a lot of time (e.g., 600 μs on 32 processors), but also because asynchronously operating PEs can overlap different computation phases (cf. Fig. 3).

5.5. Variants of iterative methods. Table 7 details some other variants of iterative solvers in PILS. For all these experiments, the linear system DR15C was solved to a relative residual norm of 10^{-10}. We compare execution times on 64 processors of an Intel Paragon and a Fujitsu AP1000.

Our experience on other machines has shown that Bi-CGSTAB [38], split preconditioned with a D-ILU factorization, is the fastest iterative method on most of the linear systems occurring in semiconductor device simulation. It appears to be the fastest solver in our DMPP version as well; this is why we chose split D-ILU preconditioned Bi-CGSTAB for all other measurements in this article, and as a reference for the following comparison of other methods. The present section just highlights a few aspects of the methods that appear to be relevant for the parallelization; the interested reader is referred to [27] for further details.

BiCG [11] requires transposed matrix–vector products, which is often an impediment in a parallel implementation. The good speedup here is due to the partitioning by matchings in the jagged diagonal data structure of PILS, and to remote updates in the code generated by Oxygen. Conjugate Gradients Squared (CGS) [36] has a very similar structure to Bi-CGSTAB, which explains the similarity in the parallelization results. GMRES(∞) [34] requires a minimum number of matrix–vector multiplications, at the expense of memory and other operations. In particular, GMRES requires many dot products of vectors. These global reduction operations are a well-known bottleneck in parallel implementations. High message latencies make this bottleneck more visible on the Paragon. The dot-product bottleneck can be reduced by grouping more global reduction operations together or by restarting the method every few iterations (10 in our experiment), but most of these kinds of variants of GMRES appear to lose some numerical stability and convergence speed, and this can be devastating on ill-conditioned problems.

The most popular preconditioners perform an approximate factorization of the original system matrix A into a product of two sparse triangular matrices $A \approx LU$. If $L + U$ has the same sparsity structure as A, a preconditioned matrix–vector product requires twice as many computations as an unpreconditioned product. Our favorite preconditioner is D-ILU, a variant of incomplete factorizations in which all off-diagonal entries of the triangular factors coincide (modulo a diagonal multiplier) with the off-diagonal nonzeros of the original matrix. When D-ILU is used in split position (that is, every preconditioned matrix–vector product applies the operator $L^{-1}AU^{-1}$ to a vector), Eisenstat's trick [8] saves almost half of the computations for a preconditioned matrix–vector product. The savings are only one quarter in right or left preconditioning (using the operators $AU^{-1}L^{-1}$ or $U^{-1}L^{-1}A$, respectively). Classical ILU preconditioning [25], [9] modifies all nonzero entries on unstructured matrices as ours, so that Eisenstat's savings cannot be refunded. The SSOR preconditioner has the same general structure as D-ILU, and thus achieves the same parallelization effect.

Note that the most powerful preconditioner in PILS, an approximate factorization with a numerical threshold, is too finely grained for an efficient parallelization on today's DMPPs. In some large device simulations, a few among several hundreds of linear systems are so ill conditioned that they require this more robust preconditioner. This preconditioner runs mostly in scalar mode on vector computers. In our programming paradigm, it can run sequentially on the controller. A fully distributed parallelization would be severely restricted in this case.

Nested iterative solvers are another alternative for very ill conditioned linear systems that resist standard preconditioning. A preconditioned iterative method (Bi-CGSTAB or GMRES in our examples) is used to precondition another iterative method (Generalized Conjugate Residuals (GCR) in the examples). The inner method is again preconditioned by split D-ILU. The first of the nested solvers listed in Table 7 is equivalent to GMRESR(10) [39]; it uses ten inner iterations for each outer iteration. The inner method of our second nested solver iterates only to a relative tolerance of 10^{-2} or terminates if this inner tolerance cannot be achieved within ten iterations.

6. Conclusions. With the results of §5, and at least for packages similar to PILS, i.e., applications based on finite elements and unstructured sparse matrix computations, we can now answer the three main questions asked at the beginning of this project:

- How difficult is the migration of existing vector supercomputer application codes to DMPPs?
- How much can compilers ease this migration?
- How should applications be designed to keep portability among vector and parallel architectures?

In our case, the migration was possible due to several features of both the application code and the supporting parallelizing compiler.

The clear separation of Fortran code for numerically intensive computing and C++ code for the more complex control-flow decisions was a key feature of PILS allowing us to successfully use the parallelization strategy outlined in §4.4. The fact that two different languages are used to implement both parts is not important; what allowed us to reduce host/PE traffic significantly was the relatively few data dependencies between the two code parts. PILS has been used in several applications, including some semiconductor device simulators. In these applications it was used as a library; i.e., the surrounding application used PILS library calls to solve a linear system of equations stored in memory. The main lesson for the user of similar packages (already optimized for vector supercomputers) is to access main data structures if possible exclusively using library calls provided by the underlying package. The less data traffic exists between the surrounding application and the library, the easier our parallelization strategy could be applied. On the other hand, the package developer should include a set of library routines that cover all reasonable uses of the data, so that uncontrolled direct access to the main data structures can be avoided.

Several features of Oxygen are crucial for the efficient parallelization of PILS: (1) general global name space support at run-time, (2) support of remote fetches and *remote updates* of distributed data (i.e., no "owner computes"), (3) replication of sequential parts of the program execution, and (4) avoidance of global synchronizations by synchronizing processors only locally through the use of local communications as required by data exchanges. None of the other compilation systems listed in §3.3 supports all four of these features.

The general opinion of the DMPP operating system research and development community seems to favor a hostless programming paradigm. We agree that in contrast to host-based machines like the AP1000, on machines like the Paragon, this paradigm offers more elegance and potential to overcome the host I/O bottleneck. However, unless full I/O support is provided on at least some selected PEs of the system (including a large physical and even larger virtual memory), a strategy as outlined in §4.4 for the parallelization of PILS could not be applied to such a hostless system. On the Paragon, which supports virtual memory on all PEs, the lack of physical memory on the controlling PE accounts for longer execution times when solving large problems.

In addition, as we have already done in [2], [3], we must again advocate *balanced* DMPP architectures. Applications like PILS can only be parallelized efficiently on DMPPs if the underlying architecture supports performing computation and communication engines equally well. For host-based DMPPs, the host can be incorporated into the overall parallel application, and thus host communication speed is equally essential. The communication time for message sizes required by a given application should be of the same order of magnitude as the time to consume the message contents in computations. As shown in Fig. 4, the compiler generated parallel code with many small-message exchanges between processors (due to application requirements, not due to compiler restrictions). Therefore, the efficient execution of PILS depends strongly on the communication startup latency featured by the underlying DMPP architecture. A DMPP's communication latency, its communication bandwidth, and its local computation performance should be balanced to support efficient parallel execution of applications similar to PILS.

Acknowledgments. Access to AP1000 machines of various sizes was provided by the Fujitsu Parallel Computing Research Facilities in Kawasaki, Japan. We would like to thank T. Horie from Fujitsu Laboratories Ltd. for his support. We are grateful to M. Vollmer from Intel SSD for providing us access to the above-described Paragon system in Beaverton, OR. We thank W. M. Coughran, K. Decker, and W. Sawyer for their careful proofreading.

REFERENCES

[1] M. Annaratone, M. Fillo, M. Halbherr, R. Rühl, P. Steiner, and M. Viredaz, *The* K2 *distributed memory parallel processor: Architecture, compiler, and operating system*, in Proc. Supercomputing 91, Albuquerque, NM, ACM-IEEE, 1991.

[2] M. Annaratone, C. Pommerell, and R. Rühl, *Interprocessor communication speed and performance in distributed-memory parallel processors*, in Proc. 16th Symposium on Computer Architecture, Jerusalem, IEEE-ACM, 1989, pp. 315–324.

[3] M. Annaratone and R. Rühl, *Balancing interprocessor communication and computation on torus-connected multicomputers running compiler-parallelized code*, in Proc. SHPCC 92, Williamsburg, VA, IEEE, 1992.

[4] APR (Applied Parallel Research), *HPF parallelization tools for clustered workstations and distributed memory multi-processor systems*, HPC Select News, article 914, Aug. 1993. Further information available from APR, Placerville, CA.

[5] H. Berryman, J. Saltz, and J. Scroggs, *Execution time support for adaptive scientific algorithms on distributed memory machines*, Concurrency: Practice and experience, 21 (1991), pp. 137–144.

[6] C. Chase and A. Reeves, *Data remapping for distributed-memory multicomputers*, in Proc. Scalable High Performance Computing Conference, IEEE Computer Society Press, Piscataway, NJ, 1992, pp. 137–144.

[7] R. Das, J. Saltz, and R. Hanxleden, *Slicing analysis and indirect access to distributed arrays*, in Proc. 6th Workshop on Languages and Compilers for Parallel Computing, Portland, OR, Lecture Notes in Computer Science vol. 768, Springer-Verlag, Berlin, New York, 1993.

[8] S. C. Eisenstat, *Efficient implementation of a class of preconditioned conjugate gradient methods*, SIAM J. Sci. Statist. Comput., 2 (1981), pp. 1–4.

[9] H. C. Elman, *Iterative Methods for Large, Sparse Nonsymmetric Systems of Linear Equations*, Ph.D. thesis, Yale University Department of Computer Science, Apr. 1982.

[10] J. Field, G. Ramalingam, and F. Tip, *Parametric program slicing*, in Proc. POPL 95, 22nd Symposium Principles of Programming Languages, ACM SIGPLAN-SIGACT, 1994.

[11] R. Fletcher, *Conjugate gradient methods for indefinite systems*, in Proc. 6th Biennial Dundee Conference on Numerical Analysis, G. A. Watson, ed., Springer-Verlag, Berlin, New York, 1976.

[12] H. Gantt, *Organizing for work*, Industrial Management, (1919), pp. 89–93.

[13] M. Gupta and P. Banerjee, *PARADIGM: A compiler for automatic data distribution on multicomputers*, in Proc. ACM International Conference on Supercomputing, Tokyo, Japan, 1993.

[14] P. Hatcher, M. Quinn, R. Anderson, A. Lapadula, B. Seevers, and A. Bennett, *Architecture-independent scientific programming in Dataparallel C: Three case studies*, in Proc. Supercomputing 91, Albuquerque, NM, ACM-IEEE, 1991.

[15] M. T. Heath and J. E. Finger, *ParaGraph: A Tool for Visualizing Performance of Parallel Programs*, User Guide, University of Illinois at Urbana-Champaign (UIUC)/Oak Ridge National Laboratory, 1993.

[16] G. Heiser, C. Pommerell, J. Weis, and W. Fichtner, *Three dimensional numerical semiconductor device simulation: Algorithms, architectures, results*, IEEE Trans. Comput.-Aided Design Integrated Circuits, 10 (1991), pp. 1218–1230.

[17] S. Hirandani, K. Kennedy, and C. Tseng, *Compiler optimizations for Fortran D on MIMD distributed-memory machines*, in Proc. Supercomputing 91, Albuquerque, NM, ACM-IEEE, 1991.

[18] HPFF (High Performance Fortran Forum), *High Performance Fortran Language Specification: Version* 1.0, Sci. Programming, 2 (1993), pp. 1–170.

[19] Intel Corporation Supercomputers Division, *Paragon (TM) XP/E Supercomputer Specification sheet Order No.* 910-001 11/93/5K/JP, Intel Corporation Supercomputers Division, Portland, OR, 1993.

[20] H. Ishihata et al., *An architecture of highly parallel computer* AP1000, in Proc. Pacific Rim Conference on Communications, Computers and Signal Processing, IEEE, 1991, pp. 13–16.

[21] C. Koelbel and P. Mehrotra, *Compiling global name-space parallel loops for distributed execution*, IEEE Trans. Parallel Distrib. Systems, 2 (1991), pp. 441–451.

[22] C. Koelbel, P. Mehrotra, and J. V. Rosendale, *Supporting shared memory data structures on distributed memory architectures*, in Proc. Second Symposium on Principles and Practice of Parallel Programming, Seattle, WA, ACM SIGPLAN, 1990, p. 177.

[23] C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh, *Basic linear algebra subprograms for Fortran usage*, ACM Trans. Math. Software, 5 (1979), pp. 308–323.

[24] J. Li and M. Chen, *Compiling communication-efficient programs for massively parallel machines*, IEEE Trans. Parallel Distrib. Systems, 2 (1991), pp. 361–375.

[25] J. A. Meijerink and H. A. van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comput., 31 (1977), pp. 148–162.

[26] M. Neeracher and R. Rühl, *Automatic parallelization of LINPACK routines on distributed memory parallel processors*, in Proc. IEEE International Parallel Processing Symposium, New Port Beach, CA, IEEE, 1993.

[27] C. Pommerell, *Solution of Large Unsymmetric Systems of Linear Equations*, Hartung-Gorre Verlag, Konstanz, Germany, 1992.

[28] C. Pommerell, M. Annaratone, and W. Fichtner, *A set of new mapping and coloring heuristics for distributed-memory parallel processors*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 194–226.

[29] C. Pommerell and W. Fichtner, *PILS: An iterative linear solver package for ill-conditioned systems*, in Proc. Supercomputing 91, Albuquerque, NM, ACM-IEEE, 1991, pp. 588–599.

[30] ———, *Memory aspects and performance of iterative solvers*, SIAM J. Sci. Comput., 15 (1994), pp. 460–473.

[31] R. Rühl, *Evaluation of compiler generated parallel programs on three multicomputers*, in Proc. ACM International Conference on Supercomputing, Washington, DC, ACM, 1992.

[32] R. RÜHL, *A Parallelizing Compiler for Distributed-Memory Parallel Processors*, Hartung-Gorre Verlag, Konstanz, Germany, 1992.

[33] Y. SAAD, *Krylov subspace methods on supercomputers*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 1200–1232.

[34] Y. SAAD AND M. H. SCHULTZ, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[35] H. D. SIMON, *Partitioning of unstructured problems for parallel processing*, Comput. Systems Engrg., 2 (1991), pp. 135–148.

[36] P. SONNEVELD, *CGS, a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 36–52.

[37] B. STROUSTRUP, *The C++ Programming Language*, Addison-Wesley, Reading, MA, 1987.

[38] H. A. VAN DER VORST, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 631–644.

[39] H. A. VAN DER VORST AND C. VUIK, *GMRESR: A family of nested GMRES methods*, Technical rep. 91-80, Dept. of Techn. Math. and Inf., TU-Delft, 1991.

[40] J. WU, J. SALTZ, S. HIRANDANDANI, AND H. BERRYMAN, *Runtime compilation methods for multicomputers*, Proc. 1991 International Conf. on Parallel Processing, II, Software, CRC Press, Aug. 1991, pp. II–26–30.

[41] H. P. ZIMA, H. J. BAST, AND M. GERNDT, *SUPERB: A tool for semi-automatic SIMD/MIMD parallelization*, Parallel Comput., 6 (1988), pp. 1–18.

SIAM J. SCI. COMPUT.
Vol. 17, No. 1, pp. 260–262, January 1996

A SIMPLE PARALLEL ALGORITHM FOR POLYNOMIAL EVALUATION*

LEI LI†, JIE HU‡, AND TADAO NAKAMURA‡

Abstract. In this paper, we show a simple parallel algorithm for polynomial evaluation. By this method, we only need $2N/p + \log_2 p$ steps on p processors (where $p \leq O(N^{1/2})$) to evaluate a polynomial of degree N on an SIMD computer or an MIMD computer, which is a decrease of $\log_2 p$ steps as compared with the p-order Horner method [S. Lakshmivarahan and S. K. Dhall, *Analysis and Design of Parallel Algorithms*, McGraw-Hill, New York, 1990], and also a decrease of $(2 \log_2 p)^{1/2}$ steps as compared with some other algorithms on an MIMD computer [J. I. Munro and M. Paterson, *J. Comput. System Sci.*, 7 (1973), pp. 189–198, K. Maruyama, *IEEE Trans. Comput.*, C-22 (1973), pp. 2–5]. The new algorithm is simple in structure and easy to implement.

Key words. parallel algorithm, polynomial evaluation

AMS subject classification. MR65

1. Introduction.

As a basic subroutine, the polynomial evaluation of degree N is often used in many parallel iterative computations. For an SIMD computer, Estrin's algorithm [2] needs $T = 2(\log_2 N + 1)$ steps when using $p = [N/2]$ processors; here $[x]$ is an integer satisfying the inequality $x \leq [x] < x+1$. And the p-order Horner method needs $T = 2N/p + 2 \log_2 p$ steps when using p processors [1]. For an MIMD computer, the number of steps can be further decreased but the algorithms become extremely complex to implement. For example, Munro and Paterson [3] and Maruyama [4] proved that $p = N$, $T \leq \log_2 N + (2 \log_2 N)^{1/2} + O(1)$, and when $p = N^{1/2}$, $T \leq 2N/p + \log_2 p + (2 \log_2 p)^{1/2} + O(1)$. Muraoka applied the method of folding and realized $T \leq 1.44 \log_2 N + O(1)$ when using N processors [1]. Even when there is an infinite number of processors, Kosaraju [5] derived a lower bound for $T(N)$ by giving the inequality $T(N) \geq \log_2 N + (2 \log_2 N)^{1/2} - (\log_2 N)^{1/4} - C$, where the constant $C > 0$.

In this paper, we present a simple parallel algorithm for evaluating a polynomial of degree N on an SIMD computer or an MIMD computer. This algorithm needs $T \leq 2N/p + \log_2 p$ steps when using $p \leq O(N^{1/2})$ processors.

2. Algorithm.

Let $f(x)$ be a polynomial of degree N,

$$f(x) = \sum_{i=0}^{N} a_i x^i, \tag{1}$$

and $N + 1 = KL$. Suppose the number of processors $p = L + 1$. First we divide the $N + 1$ items of $f(x)$ into L groups; i.e.,

$$f(x) = \sum_{i=0}^{L-1} b_i x^{iK},$$

where

$$b_i = a_{iK} + a_{iK+1}x + \cdots + a_{iK+K-1}x^{K-1} \quad \text{for } i = 0, 1, \ldots, L - 1. \tag{2}$$

Therefore the new algorithm can be obtained as follows:

*Received by the editors June 29, 1994; accepted for publication (in revised form) December 23, 1994.

†Department of Information System Engineering, Aomori University, Aomori, 030 Japan (lilei@auvax.aomori-u.ac.jp).

‡Graduate School of Information Science, Tohoku University, Sendai, 980 Japan (tadao@archi.is.tohoku.ac.jp).

(a) First, compute $b_0, b_1, \ldots, b_{L-1}$ in parallel using L processors by the vector Horner method. We know $2(K-1)$ steps are needed.

(b) Next, serially compute $x^K, x^{2K}, \ldots, x^{(L-1)K}$.

Now, let us show how to compute each x^{iK} by the fast power method. Suppose $2^r \leq K < 2^{r+1}$; then K can be expressed in binary form:

$$K = d_0 + d_1 * 2 + d_2 * 2^2 + \cdots + d_r * 2^r,$$

where $d_i \in (0, 1), i = 0, 1, \ldots, r-1, d_r = 1$. Then x^K can be written as follows:

$$x^K = x^{d0}(x^{d1}(x^{d2} \ldots (x^{dr-1}(x^{dr})^2)^2 \ldots)^2)^2; \tag{3}$$

therefore x^K can be computed serially in $2r(\leq 2[\log_2 K])$ steps using one processor. Furthermore using $L-2$ multiplications, we can get

$$x^{iK} \quad \text{for } i = 2, 3, \ldots, L-1. \tag{4}$$

(c) And last, compute the inner product

$$b_0 + b_1 x^K + b_2 x^{2K} + \cdots + b_{L-1} x^{(L-1)K} \tag{5}$$

in parallel using L processors. We know that at most $1 + [\log_2 L]$ steps are needed.

3. The analysis of time complexity. Now let us turn to analyze the number of steps of the new algorithm for an MIMD computer. Note that expressions (2)–(4) can be computed in the following parallel way. We compute (2) by using L processors, and compute (3) and (4) by using one processor in parallel. Finally we can compute (5) by using L processors. Thus, the whole process can be computed in parallel in

$$\begin{aligned} T &\leq \max(2K-2, 2\log_2 K + L - 2) + 2 + \log_2 L \\ &= \begin{cases} 2K + \log_2 L & \text{when } L \leq 2K - 2\log_2 K, \\ L + \log_2 K + \log_2(N+1) & \text{when } L > 2K - 2\log_2 K \end{cases} \end{aligned}$$

steps. Let

$$Y_1 = f(L) = 2K + \log_2 L = 2(N+1)/L + \log_2 L \quad (\text{where } L \leq 2K - 2\log_2 K);$$

then the derived function of Y_1 is

$$\begin{aligned} Y_1' &= -2(N+1)/L^2 + 1/\ln 2 * 1/L \\ &= 1/L * (\log_2 e - 2(N+1)/L) < 0. \end{aligned}$$

Thus function Y_1 is monotone decreasing. Similarly, let

$$\begin{aligned} Y_2 = g(L) &= L + \log_2 K + \log_2(N+1) \\ &= L - \log_2 L + 2\log_2(N+1) \quad (\text{where } L > 2K - 2\log_2 K); \end{aligned}$$

then the derived function of Y_2 is

$$Y_2' = 1 - \log_2 e * 1/L > 0,$$

and so Y_2 is monotone increasing. Therefore the number of steps takes the minimum $T = 2K + \log_2 L$ when $L = 2K - 2\log_2 K$. Now we can get the following conclusion.

THEOREM 1. *For an MIMD computer, a polynomial of degree N(where $N+1 = KL$) can be computed in $T = 2K + \log_2 L$ steps at most when using $L+1$ processors, where $L \leq 2K - 2\log_2 K$. And when $L = 2K - 2\log_2 K$, T takes the minimum.*

For an SIMD computer, it is easy to use our method. When $p_0, p_1, \ldots, p_{L-1}$, then do addition. We can let p_L do addition by plus zero. So, we obtain an SIMD algorithm for polynomial evaluation with $T \leq 2K + \log_2 L$, $L \leq K - 2\log_2 K$. Therefore, we can get the following:

$$\text{For an SIMD computer,} \quad T \leq 2N/p + \log_2 p, \quad p < O(N^{1/2}).$$

$$\text{For an MIMD computer,} \quad T \leq 2N/p + \log_2 p, \quad p \leq O(N^{1/2}).$$

4. Conclusion. A simple parallel algorithm for evaluating a polynomial of degree N is presented in this paper. This method needs $2N/p + \log_2 p$ parallel steps on $p \leq O(N^{1/2})$ processors and has decreased by $\log_2 p$ steps and $(2\log_2 p)^{1/2}$ steps as compared with the p-order Horner method [1] and some MIMD methods [3], [4], respectively. The new method is also simple on structure and easy to implement.

REFERENCES

[1] S. LAKSHMIVARAHAN AND S. K. DHALL, *Analysis and Design of Parallel Algorithms*, McGraw-Hill Publishing Company, New York, 1990, pp. 254–273.

[2] W. S. DORN, *Generalizations of Horner's rule for polynomial evaluation*, IBM J. Res. Develop., 6 (1962), pp. 239–245.

[3] J. I. MUNRO AND M. PATERSON, *Optimal algorithms for parallel polynomial evaluation*, J. Comput. System Sci., 7 (1973), pp. 189–198.

[4] K. MARUYAMA, *On the parallel evaluation of polynomials*, IEEE Trans. Comput., C-22 (1973), pp. 2–5.

[5] S. R. KOSARAJU, *Parallel Evaluation of Division-Free Arithmetic Expressions*, Proc. Symposium on Theory of Computing, 1986, pp. 231–239.

SIAM J. SCI. COMPUT.
Vol. 17, No. 1, pp. 263–274, January 1996

018

A BLOCK QMR METHOD FOR COMPUTING MULTIPLE SIMULTANEOUS SOLUTIONS TO COMPLEX SYMMETRIC SYSTEMS*

WILLIAM E. BOYSE† AND ANDREW A. SEIDL†

Abstract. The solution of complex symmetric indefinite systems of equations where multiple solutions are required is considered. The quasi-minimum residual (QMR) method, ideally suited for these matrices, is generalized using the block Lanczos algorithm to solve multiple solutions simultaneously. This modification alone is shown, through numerical examples involving large sparse matrices from finite element discretization of Maxwell's equations, to accelerate the convergence by a factor almost as great as the number of simultaneous solutions. A natural convergence criterion for this method is presented that is shown to be as effective as, and easier to compute than, the usual equation residual. Finally, a numerical comparison of the classical incomplete Cholesky and a variant of the ILU(T) preconditioners is given showing superior performance by the latter.

Key words. QMR, complex symmetric matrices, block Lanczos algorithm, preconditioner, Maxwell's equations

AMS subject classifications. 65F10, 65N22

1. Introduction. Electromagnetic scattering applications require the solution of large matrix equations for many right-hand side vectors. Typically these vectors correspond to the numerous angles for which the simulated radar return is desired, i.e., monostatic radar cross section (RCS). When using a boundary integral equation method (aka method of moments [12]) the system matrix is general, complex, and dense. For these systems, an LU factorization is performed and the multiple solutions are generated by forward reduction and back substitution. However, when finite element methods are used to model scattering from large three-dimensional objects [1], [7], [6] they produce matrices much too large for direct factorization.

Using the finite element method to discretize Maxwell's equations in differential form necessitates the solution of large sparse complex symmetric indefinite matrices for many right-hand side vectors using iterative methods. This is by far the dominant computational task in all such scattering codes.

Recently an iterative solver of particular merit was developed which specifically addresses the complex symmetric indefinite nature of these matrices [8]. The quasi-minimum residual (QMR) method is theoretically superior to the biconjugate gradient (BCG) method for these problems and exhibits a nearly monotonically decreasing convergence behavior.

In this paper, we generalize the QMR algorithm to solve multiple right-hand side vectors simultaneously as was done for the conjugate gradient (CG) method in [14], [15]. This involves utilization of the block rather than the point Lanczos algorithm and of course the requisite matrix rather than scalar arithmetic. For the application considered, this modification alone results in an acceleration of convergence by a factor almost as great as the number of simultaneous solutions [4]. A natural convergence criterion for this method is also presented and is shown to be equal to the usual equation residual in effectiveness.

Further convergence acceleration is achieved by using an appropriate preconditioner. Both the "classical" incomplete Cholesky preconditioner and a variant of the ILU(T) [17], [18], [3] preconditioner are implemented and evaluated on a scattering problem.

The outline of this paper is as follows. In §2, we present the block version of the QMR algorithm for complex symmetric matrices and the natural convergence criterion. Arguments are given for the potential speedup of convergence by using the block algorithm. In §3,

*Received by the editors June 29, 1994; accepted for publication (in revised form) September 20, 1994.

†Lockheed Palo Alto Research Laboratories, 3251 Hanover Street, Palo Alto, CA 94304-1191 (boyse@lmsc.lockheed.com, seidl@lmsc.lockheed.com).

numerical examples are shown illustrating the effectiveness of the natural convergence criterion and the effects of preconditioning and blocksize on the convergence rate as applied to large finite element problems. Finally, in §4, we draw some conclusions and comment on parallel applications.

2. Block QMR method. The QMR method, based on the Lanczos algorithm, has a natural generalization to the simultaneous solution of multiple right-hand sides in the block Lanczos algorithm. Converting the QMR algorithm to block form requires generalizing vectors and scalars into matrices and deriving a block QR decomposition used in the solution process. The noncommutative nature of matrices requires that all the algebra used in the derivation respect this restriction. The algorithm described and notation used follow [8] with modifications as needed.

We will call the symmetric indefinite bilinear form

$$\langle x, y\rangle_Q = \sum_k x_k y_k, \tag{1}$$

where x and y are complex vectors, the "quasi" inner product. It differs from the usual inner product in that neither constituent is conjugated. With respect to this form, a complex symmetric matrix is "quasi–self-adjoint," i.e.,

$$\langle Ax, y\rangle_Q = \langle x, Ay\rangle_Q. \tag{2}$$

This permits the formal generalization of techniques developed for real symmetric matrices to complex symmetric matrices, as in the derivation of the BCG algorithm from the CG algorithm. In the remainder of the paper, the prefix of quasi will refer to an operation with respect to the bilinear form in (1).

The block QMR algorithm used addresses the solution of a system of equations involving a complex symmetric indefinite matrix

$$\mathbf{A}x = b, \tag{3}$$

where **A** is an n-by-n matrix, and x and b are n-by-m matrices. In this case, the m columns of b are m right-hand side vectors and the m columns of x are their corresponding solution vectors.

2.1. Symmetric block Lanczos recursion. Let x_0 be an initial guess at the solution, which may be zero. The symmetric block Lanczos algorithm is then

Initialization:

$$r_0 = b - Ax_0 \tag{4}$$

$$v_0 = 0 \tag{5}$$

$$\tilde{v}_1 = r_0 \tag{6}$$

For k = 1, 2, 3, . . .

$$v_k\beta_k = \tilde{v}_k \tag{7}$$

$$\tilde{v}_{k+1} = Av_k - v_{k-1}\beta_k^T \tag{8}$$

$$\alpha_k = v_k^T \tilde{v}_{k+1} \tag{9}$$

$$\tilde{v}_{k+1} = \tilde{v}_{k+1} - v_k\alpha_k \tag{10}$$

where α_k and β_k are m-by-m matrices, v_k and $\tilde{v}_k$ are n-by-m matrices, and the superscript T denotes the transpose. The operation described by (7) is the quasi-QR decomposition of $\tilde{v}_k$, where $\tilde{v}_k$ is written as a quasi-unitary matrix v_k times an upper triangular matrix β_k. This decomposition is computed using the modified Gram–Schmidt process using the quasi inner product. The Lanczos algorithm is due to Pissanetzky [16] and differs from [8] in that the computation of α_k in (9) is performed after the partial orthogonalization of Av_k in (8).

The matrix V_k is defined to be the aggregate of the columns of the individual v_j's for $j = 1, \dots, k$,

$$V_k = \begin{bmatrix} v_1 & v_2 & \dots & v_k \end{bmatrix}, \tag{11}$$

and the $k+1$-by-k block tridiagonal matrix $\tilde{T}^k$ is defined by

$$\tilde{T}_k = \begin{bmatrix} \alpha_1 & \beta_2^T & & & \\ \beta_2 & \alpha_2 & & & \\ & & \ddots & & \\ & & & \alpha_{k-1} & \beta_k^T \\ & & & \beta_k & \alpha_k \\ & & & & \beta_{k+1} \end{bmatrix}. \tag{12}$$

The Lanczos algorithm thus provides the block tridiagonal representation of the matrix A in terms of V_k,

$$AV_k = V_{k+1}\tilde{T}_k, \tag{13}$$

where the columns of V_k are a quasi-orthonormal basis for the Krylov subspace generated by all the right-hand side vectors

$$\mathrm{K}_k = \mathrm{Span}\{A^j \operatorname{col}_i(b - Ax_0) | 0 \le j \le k-1, 1 \le i \le m\}, \tag{14}$$

where $\operatorname{col}_i(b)$ is the ith column of b.

2.2. Block QMR algorithm. At iteration k we seek an approximate solution x_k of the form

$$x_k = x_0 + V_k z_k, \tag{15}$$

where z_k is a $k \cdot m$-by-m complex matrix. That is, each column of $x_k - x_0$ lies in the Krylov subspace K_k (14).

The equation residual for this approximate solution is given by

$$\begin{aligned} r_k &= b - Ax_k & (16)\\ &= b - Ax_0 - AV_k z_k & (17)\\ &= r_0 - AV_k z_k & (18)\\ &= v_1\beta_1 - V_{k+1}\tilde{T}_k z_k. & (19) \end{aligned}$$

Let Ω_k be a $k \cdot m$-by-$k \cdot m$ block diagonal weight matrix

$$\Omega_k = \mathrm{Diag}(\omega_1, \omega_2, \dots, \omega_k), \tag{20}$$

where ω_i is an m-by-m matrix to be defined later. The residual may then be written

$$r_k = V_{k+1}\Omega_{k+1}^{-1}\left[\omega_1 e_1 \beta_1 - \Omega_{k+1}\tilde{T}_k z_k\right], \tag{21}$$

where e_1 is the n-by-m matrix which is zero except for the first m rows, which are the m-by-m identity matrix. This represents each column of the residual matrix as a linear combination of the columns of $V_{k+1}\Omega_{k+1}^{-1}$.

Suppose for a moment that all quantities are in fact real, that $m = 1$, and that Ω_{k+1} is the identity matrix. Then V_k would be a unitary matrix, not just quasi-unitary. The minimum residual solution could then be obtained by minimizing the Euclidean norm of the coefficients of the columns of V_k, i.e., minimizing

$$\|\omega_1 e_1 \beta_1 - \Omega_{k+1}\tilde{T}_k z_k\|. \tag{22}$$

For the complex symmetric case, the QMR algorithm mimics this solution by also minimizing these coefficients, where the block diagonal weighting matrix Ω_{k+1} is chosen so that the columns of $V_{k+1}\Omega_{k+1}^{-1}$ are each of unit Euclidean norm. In light of (20) this may be accomplished by defining

$$\omega_k \quad = \quad \mathrm{Diag}(\|\mathrm{col}_i(v_k)\|). \tag{23}$$

The block QMR algorithm chooses z_k so that

$$\|\mathrm{col}_i(\omega_1 e_1 \beta_1 - \Omega_{k+1}\tilde{T}_k z_k)\| \tag{24}$$

is a minimum independent for $i = 1, \ldots, m$. The natural convergence criterion for this method is the maximum of the minima attained in (24) and is termed the quasi residual.

This minimization is accomplished using the QR decomposition [11], where a unitary matrix Q_{k+1} is determined so that

$$\begin{aligned} Q_{k+1}\Omega_{k+1}\tilde{T}_k \quad &= \quad \begin{bmatrix} R_k \\ 0 \end{bmatrix} \\ &= \quad \begin{bmatrix} \zeta_1 & \eta_2 & \theta_3 & & \\ & \zeta_2 & \eta_2 & \ddots & \\ & & \ddots & \ddots & \theta_k \\ & & & \ddots & \eta_k \\ & & & & \zeta_k \\ & & & & 0 \end{bmatrix}, \end{aligned} \tag{25}$$

where R_k is upper triangular as shown. This factorization is stable for indefinite matrices under consideration.

The matrix Q_{k+1} is updated from the previous iteration by setting

$$Q_{k+1} = \begin{bmatrix} I_{(k-1)m} & 0 \\ 0 & Q(a_k, b_k, c_k, d_k) \end{bmatrix} \begin{bmatrix} Q_k & 0 \\ 0 & I_m \end{bmatrix}, \tag{26}$$

where

$$Q(a_k, b_k, c_k, d_k) = \begin{bmatrix} a_k & b_k \\ c_k & d_k \end{bmatrix} \tag{27}$$

is a $2m$-by-$2m$ unitary matrix written as four m-by-m blocks, and I_m is the m-by-m identity matrix.

A problem equivalent to minimizing (24) may then be written as minimizing

$$\left\|\mathrm{col}_i\left(Q_{k+1}\omega_1 e_1 \beta_1 - \begin{bmatrix} R_k \\ 0 \end{bmatrix} z_k\right)\right\| \tag{28}$$

for $i = 1, \ldots, m$. This is easily done by defining

$$\tilde{t}_{k+1} = Q_{k+1}\omega_1 e_1 \beta_1 = \begin{bmatrix} t_k \\ \tilde{\tau}_{k+1} \end{bmatrix}, \tag{29}$$

where $\tilde{\tau}_{k+1}$ is an m-by-m matrix, and setting

$$z_k = R_k^{-1}\, t_k. \tag{30}$$

The quasi residual QRES_k is computed directly from the quantity $\tilde{\tau}_{k+1}$ as

$$\text{QRES}_k = \max_i \|\text{col}_i(\tilde{\tau}_{k+1})\|. \tag{31}$$

This will be compared with the equation residual

$$\text{RES}_k = \max_i \|\text{col}_i(Ax_k - b)\| \tag{32}$$

as an equivalent convergence criterion. Note that when using these criteria all the simultaneous solutions will satisfy the convergence tolerance individually.

2.3. Block QMR algorithm. The block algorithm closely follows that given in [8]. It is worth noting that Freund's algorithm makes no use of multiplicative commutivity of scalars in its derivation and therefore resembles the current algorithm for which lack of commutivity for matrices must be respected.

Initialization:

$$\begin{aligned}
v_0 = p_0 = p_{-1} &= 0 && (33)\\
c_0 = b_{-1} = b_0 &= 0 && (34)\\
a_0 = d_{-1} = d_0 &= I_m && (35)\\
\tilde{v}_1 &= b - Ax_0 && (36)\\
v_1\beta_1 &= \tilde{v}_1 && (37)\\
\omega_1 &= \text{Diag}\{\|\text{col}_i(v_1)\|\} && (38)\\
\tilde{\tau}_1 &= \omega_1\beta_1 && (39)
\end{aligned}$$

For k = 1, 2, 3, . . .

$$\begin{aligned}
\tilde{v}_{k+1} &= Av_k - v_{k-1}\beta_k^T && (40)\\
\alpha_k &= v_k^T\tilde{v}_{k+1} && (41)\\
\tilde{v}_{k+1} &= \tilde{v}_{k+1} - v_k\alpha_k && (42)\\
v_{k+1}\beta_{k+1} &= \tilde{v}_{k+1} && (43)\\
\omega_{k+1} &= \text{Diag}\{\|\text{col}_i(v_{k+1})\|\} && (44)\\
\theta_k &= b_{k-2}\omega_{k-1}\beta_k^T && (45)\\
\eta_k &= a_{k-1}d_{k-2}\omega_{k-1}\beta_k^T + b_{k-1}\omega_k\alpha_k && (46)\\
\tilde{\zeta}_k &= c_{k-1}d_{k-2}\omega_{k-1}\beta_k^T + d_{k-1}\omega_k\alpha_k && (47)
\end{aligned}$$

Next, a $2m$-by-$2m$ unitary matrix $Q(a_k, b_k, c_k, d_k)$ is computed such that

$$Q(a_k, b_k, c_k, d_k)\begin{bmatrix} \tilde{\zeta}_k \\ \omega_{k+1}\beta_{k+1} \end{bmatrix} = \begin{bmatrix} \zeta_k \\ 0 \end{bmatrix} \tag{48}$$

where ζ_k is upper triangular. Then,

$$p_k = (v_k - p_{k-1}\eta_k - p_{k-2}\theta_k)\,\zeta_k^{-1} \tag{49}$$

$$\tau_k = a_k\tilde{\tau}_k \tag{50}$$

$$x_k = x_{k-1} + p_k\tau_k \tag{51}$$

$$\tilde{\tau}_{k+1} = c_k\tilde{\tau}_k \tag{52}$$

Here, a_*, b_*, c_*, d_* α_*, β_*, θ_*, η_*, ζ_*, $\tilde{\zeta}_*$, ω_*, τ_*, and $\tilde{\tau}_*$ are all m-by-m matrices with ζ_* and β_* upper triangular. The terms p_*, v_*, $\tilde{v}_*$, and x_* are n-by-m matrices. Equations (37) and (43) are quasi-QR decompositions of the right-hand side matrix in each equation.

At the end of each iteration, the quasi residual is computed and compared with a given convergence limit. The equation residual RES_k may also be computed, either directly or by accumulating in another work matrix [8].

Aside from the obvious matrix arithmetic and quasi-QR factorizations, the main difference between this and Freund's algorithm is the QR factorization in (48). In the point algorithm, this is just a complex Givens rotation of the form

$$Q = \begin{bmatrix} c_j & \overline{s_j} \\ -s_j & c_j \end{bmatrix}. \tag{53}$$

The generalization to the block algorithm necessitates a more general form, (48), which then affects the rest of the algorithm. This factorization is computed using a conventional QR decomposition algorithm [11] of rank $2 \cdot m$ and then partitioning the matrix into four m-by-m submatrices.

2.4. Convergence improvement. O'Leary [14] showed that the effect on the convergence rate of the block CG algorithm, with blocksize of m, was to remove the deleterious effects of the $m - 1$ smallest eigenvalues. Specifically, she showed that while the worse-case convergence rate of the point CG algorithm is determined by the condition number, i.e., ratio of largest to smallest eigenvalue, the worse-case convergence of the block algorithm is determined by the ratio of the largest to mth smallest eigenvalue, and thus is no worse and possibly much better. Theoretically, this could be of great benefit when there are a few small eigenvalues that slow the convergence rate. O'Leary also observed significant convergence acceleration of eigenvalue computations using a block algorithm and randomly generated initial vectors.

A heuristic argument for the convergence advantages of the block algorithm over the point algorithm is given by a comparison of the Krylov subspaces used in the approximations.

In the point algorithm, we have

$$x_k - x_0 \in \text{Span}\{A^j\,(b - Ax_0) | 0 \le j \le k-1\}. \tag{54}$$

Thus, the exact vector $x^* - x_0$ is approximated in a subspace of dimension k. However, in the block algorithm, we have

$$\text{col}_l(x_k - x_0) \in \text{Span}\{A^j\,\text{col}_i(b - Ax_0) | 0 \le j \le k-1, 1 \le i \le m\} \tag{55}$$

for each $l = 1, \ldots, m$. In this case, the approximating subspace has dimension $m \cdot k$, or m times as great as the corresponding point algorithm. The larger dimension of the subspace should provide a better approximation and thus more rapid convergence. The ideal limit of convergence improvement would be a reduction in the number of iterations by a factor of m, exactly corresponding to the increase in dimension of the approximating subspace.

The word "should" must be emphasized because the behavior of this algorithm in real applications is dependent on the right-hand side vectors. Just because the Krylov subspace is larger in the block rather than the point algorithm does not necessarily imply that the approximation will be better. The Krylov subspaces corresponding to the individual right-hand side vectors may not provide good approximations to solutions other than their corresponding one. In extreme cases, the right-hand sides might be linearly dependent, which would cause the presented algorithm to fail completely because the QR decomposition (37) would be singular.

While it is easy to concoct examples that will defeat the block algorithm, it is of most interest to determine what practical problems are amenable to this approach. In the next section, we will show, through a numerical example, real acceleration due entirely to the block Lanczos method as applied to the iterative solution of large finite element problems.

3. Numerical example. The application for which this technique was developed is called the hybrid finite element method (HFEM) [2], [5]. In this method, finite elements are used to discretize Maxwell's equations in differential form in and around the scatterer while a scattering integral equation is used on a smooth enclosing surface, where the fields themselves are smooth, to terminate the finite element mesh and provide the exact near field radiation condition.

The finite element implementation utilizes a scalar and vector potential formulation [1] for the electric field and second-order Lagrangian tetrahedral elements. It is essentially a 3-D vector Helmholtz formulation. The mesh is terminated on a surface of revolution where the integral equation is applied to ensure a radiating solution [13].

The discretization of the integral equation represents the equivalent electric $\vec{J} = \hat{n} \times \vec{H}$ and magnetic $\vec{M} = -\hat{n} \times \vec{E}$ currents using 1-D Hermite-cubic finite elements along the generator and Fourier modes azimuthally for each of the two orthogonal polarizations on the surface, where $\hat{t}$ is directed along the generator from pole to pole, and $\hat{\phi}$ is directed azimuthally. These basis functions are then nearly orthogonal, with only basis functions of the same polarization, same azimuthal mode, and belonging to the same Hermite element having a nonzero inner product.

In this application, the computational task is to solve the finite element boundary value problem, where $\hat{n} \times \vec{E} = -\vec{M}_k$, on the surface of revolution for each of the basis functions $\vec{M}_k$ representing the magnetic current on this surface. Once these multiple solutions are generated, the scattering problem is reduced to one involving only the efficient boundary integral equation.

The scatterer used for this example is a 10λ-by-10λ-by-$1/10\lambda$ perfectly conducting plate, where λ is the wavelength in a vacuum, enclosed in a surface of revolution resembling a thick traditional (round) pizza crust. The volume between the plate and the outer surface was discretized yielding 270,000 complex finite element degrees of freedom. The finite element matrix is real symmetric, since there is no lossy material present, and the right-hand side vectors are complex.

Two symmetric preconditioners are evaluated on this example. The first is the "classical" incomplete Cholesky (IC), where no fill is allowed. The second is an IC variant of the ILU(T) method [17], [18], [3], called IC(T), where the usual factorization algorithm is applied and "small" elements are dropped from each row of the factor as it is generated. Here "small" refers to size compared with the diagonal entry and was adjusted so that the "classical" IC and IC(T) factorizations have approximately the same number of nonzeros and thus require the same computer time to apply. This makes for a fair comparison of the two methods. Minimum degree ordering [10] is used for both preconditioners, which are applied symmetrically.

All timing tests were performed on a SUN Sparcstation rated at 3.8 double precision Megaflops (LINPACK). The algorithms, of course, all utilize double precision arithmetic.

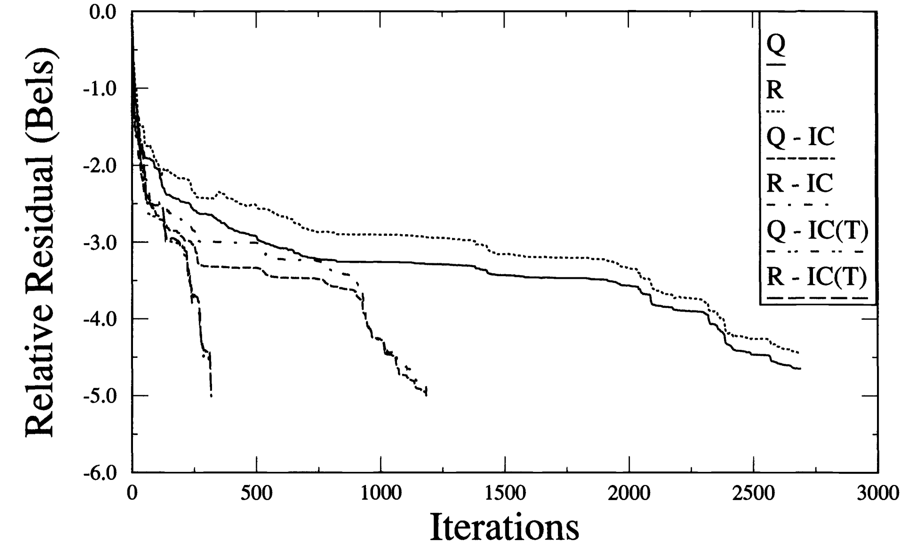


FIG. 1. *Preconditioner comparison: Quasi and equation residuals.*

While the scattering problem was too large to run to completion on the SUN, these convergence tests were completed.

Figure 1 shows the preconditioner evaluation while solving just one right-hand side vector. Convergence was defined by $Q \leq -5.0$, with iterations cut off at 1% of the rank of the matrix. The curves labeled "Q*" are the relative quasi residual while those labeled "R*" are the relative equation residual, both in Bels (the base 10 logarithm of the residuals),

$$\begin{aligned} Q &= \log_{10}\left|\frac{\text{QRES}_k}{\text{QRES}_0}\right|, \\ R &= \log_{10}\left|\frac{\text{RES}_k}{\text{RES}_0}\right|. \end{aligned} \tag{56}$$

The curves labeled Q and R show the convergence without preconditioning, Q-IC and R-IC show the convergence using the IC preconditioner, while Q-IC(T) and R-IC(T) show those using the IC(T) preconditioner.

Note that the quasi residuals are all slightly below but closely follow the equation residual. This is typical behavior experienced in running this code over the last four years. In practice, the quasi residual provides a reliable convergence criterion and eliminates both the computation and storage [8] needed to monitor the usual equation residual.

The case without preconditioning was terminated at 2700 iterations without quite reaching the -5.0 convergence criterion (-4.65). The case with IC preconditioning converged in 1184 iterations. Although this is less than half the number of iterations required with no preconditioning, each iteration is twice as expensive. A preconditioner, whose number of nonzeros is the same as the original matrix, must reduce the number of iterations by a factor of 2 just to break even, which the IC preconditioner barely achieved.

However, the IC(T) preconditioning converged in only 318 iterations. This is approximately one-ninth of the number with no preconditioning and reduced the wall clock time to solution by a factor greater than 4. Again, this is the typical behavior these preconditioners have exhibited on a wide range of problems, and the IC(T) is the only one used in practice.

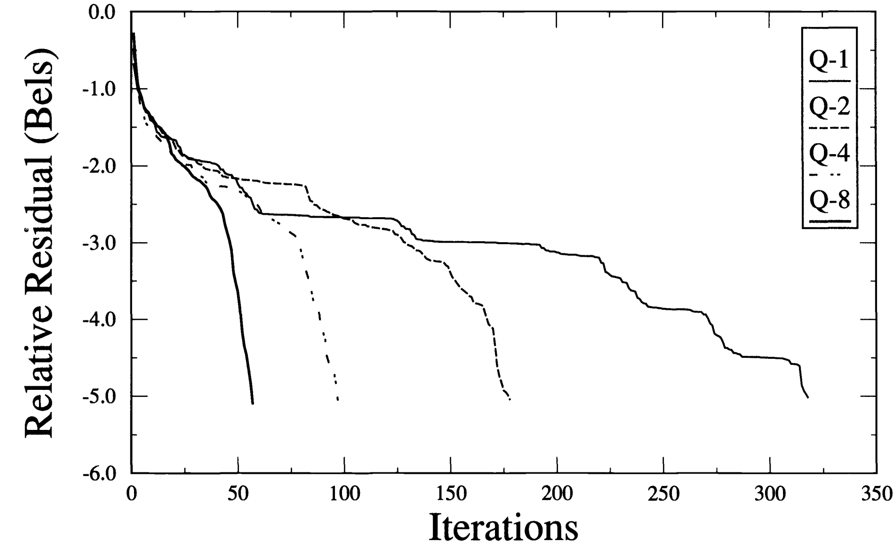


FIG. 2. *Blocksize comparison.*

Figure 2 shows the effects of varying the number of simultaneous solutions. Only the quasi residuals are shown, since they are the criterion used in practice; preconditioning using the IC(T) method and blocksizes of 1, 2, 4, and 8 are compared.

The case for 1 solution, the same as curve "Q-IC(T)" in Fig. 1, shows convergence in 318 iterations. Solving for 2 solutions simultaneously required 178 iterations—57% of the iterations required for 1 solution and close to the ideal limit of 50%. Solving for 4 solutions simultaneously took 97 iterations, 30% of that required for 1 solution, compared with the ideal limit of 25%. Finally, solving 8 solutions simultaneously required only 57 iterations, or 18% of that required for one solution, compared with the ideal limit of 12.5%.

Note that in the early convergence history, 50 to 100 iterations in Fig. 2, the convergence for one solution appears better than the convergence for two simultaneous solutions. This is caused by slower convergence for right-hand side number two only and the fact that the maximum of the residuals for right-hand sides one and two is plotted. When solving multiple simultaneous solutions, the residual for any single right-hand side will never exceed the residual obtained when that right-hand side is iterated individually.

The wall clock time per solution provides another important measure of the effectiveness of this technique. Figure 3 shows the elapsed time per solution relative to the time for one solution without preconditioning. Note that the use of the classical IC preconditioner did not affect the solution time significantly. Although convergence was accelerated, it was not enough to overcome the added computation of applying the preconditioner at each iteration. In contrast to this, the IC(T) preconditioner had a major beneficial effect on solution times, as did increasing the blocksize.

It should be noted that as blocksize m increases, the relative cost of the algorithm increases due to the m-by-m and n-by-m matrix arithmetic. These operations, however, are all BLAS3 calls, which are highly optimized on most computers capable of seriously running a problem of this size.

As a second example, a perfectly conducting sphere of radius 1 meter covered with two hemispherical shells of dielectric material 1/16 meter thick was illuminated by an 80 MHz plane wave. The electrical parameters of one shell were $\epsilon_r = 2 + \imath$ and $\mu_r = 1$, while those

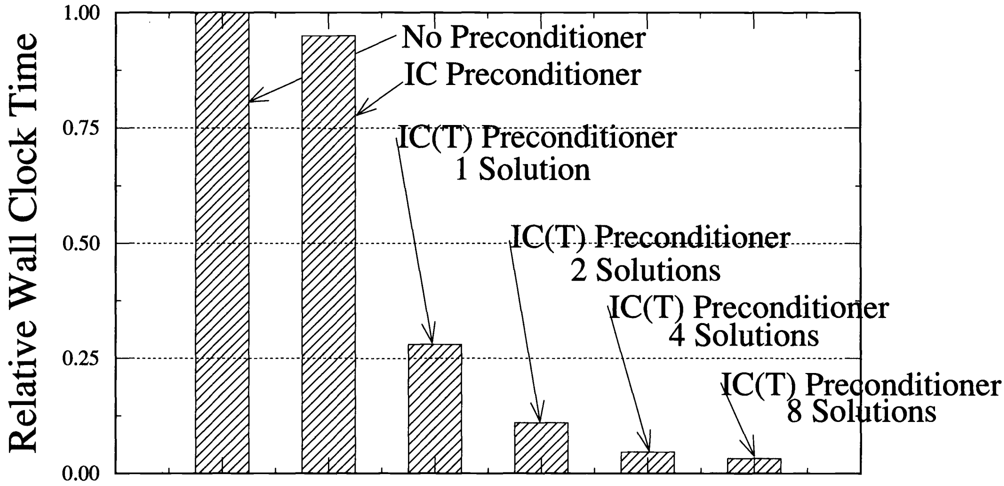


FIG. 3. *Relative solution time comparison.*

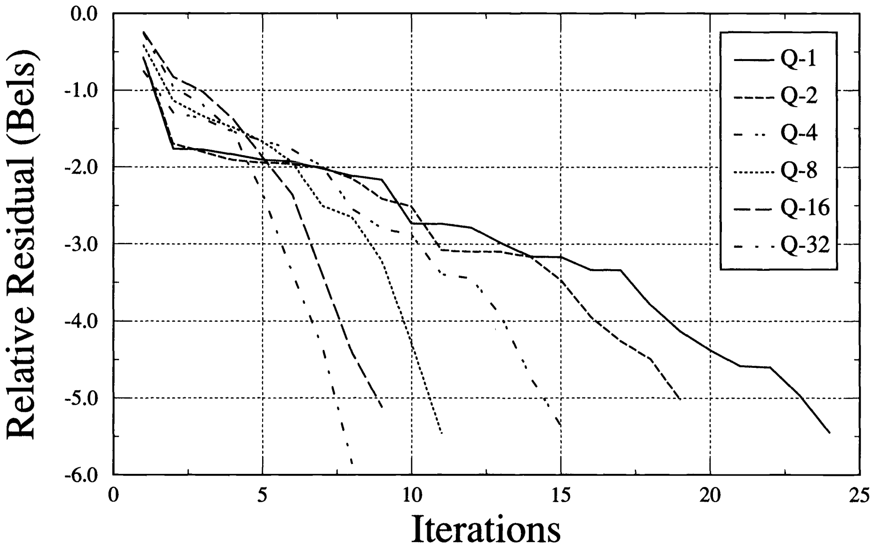


FIG. 4. *Convergence results for dielectric coated conducting sphere.*

of the other were $\mu_r = 2 + \imath$ and $\epsilon_r = 1$. The discretization of this problem yielded 10,556 unknowns. Due to the lossy materials, the matrix was complex symmetric.

Figure 4 shows the effect of varying the blocksize from 1 to 32, using the IC(T) preconditioner as usual. In this case, the convergence acceleration with increased blocksize is not as dramatic as in the previous example. This is not uncommon for problems this small where convergence is achieved in only a few iterations [3], [4].

4. Discussion. There are many topics of interest relative to this iterative method that we have not addressed. The issues of dealing with possible linear dependencies in the Krylov subspace bases, incorporating look ahead [9] into the block algorithm, and deflating the linear systems that converge before the others are not treated. Our emphasis has been on developing the basic algorithm and evaluating its behavior on our problems.

This preconditioned block QMR method has been the solver of choice for our finite element work for about four years. Applied primarily to large problems, $\sim$ 100, 000+ unknowns, it has been both reliable and effective.

Although the large example shown here uses a real matrix for computational expediency, the performance on complex matrices, generated either with impedance boundary conditions or the inclusion of lossy material, is just as good or better. In fact when the loss is removed from the material in the second example, the convergence for one right-hand side increases from 24 to 47 iterations. This behavior is typical and not unexpected.

The computation of the IC(T) preconditioner uses the algorithms given in [16], is adapted to use files on disk, and is very fast. The preconditioner used in the rank 270,000 problem took only 10 minutes elapsed time on a 3.8 MFLOP Sun workstation (compared with hours for the block QMR to complete). Since this factorization is potentially unstable for indefinite matrices, there were some initial concerns about stability. To date we have not experienced any problem attributable to instability in this calculation. We do, however, monitor the maximum and minimum pivots used in the factorization as a measure of stability.

For large problems, several factors must be balanced for optimal solution times. Increasing the blocksize accelerates convergence but also increases the memory requirements of the block QMR algorithm and increases the computational burden due to the matrix arithmetic. Computing a "less incomplete" preconditioner, by decreasing the definition of "small" in the algorithm, also accelerates convergence but increases the cost of computing and applying the preconditioner. These factors are highly dependent on the computer used and must be optimized in situ.

The dominant computational task in this method is the sparse matrix multiply and preconditioner application. In this area, too, the block algorithm shows great potential. As found in [15] and observed by us, using the block algorithm provides another dimension that may be used for parallelization.

Our preliminary work on a Cray Y-MP has shown that both the sparse matrix multiply and preconditioner applications are "embarrassingly" parallel, exhibiting a speedup of almost 8 times when solving 8 simultaneous solutions using 8 processors. This is especially significant for the preconditioner application since its recursive nature severely limits parallelization. Combining this speedup with the convergence acceleration of solving multiple simultaneous solutions means significant reductions in wall clock times per solution are possible.

While this solver performs exceedingly well in our application, the convergence acceleration due to multiple simultaneous solutions is dependent on the sparse matrix, the right-hand side vectors, and the choice of initial guesses. In what other situations block solvers will perform equally well is not yet known. Preliminary testing of a sparsely preconditioned block GMRES solver for general dense complex matrices shows convergence acceleration with increased blocksize, but not yet as much as reported here.

Acknowledgment. The authors would like to thank Noel Nachtigal for helpful discussions about this method and for his encouragement to publish this material, and the reviewers whose thoroughness greatly improved this manuscript.

REFERENCES

[1] W. E. BOYSE, D. R. LYNCH, K. D. PAULSEN, AND G. N. MINERBO, *Nodal based finite element modeling of Maxwell's equations*, IEEE Trans. Antennas Propagation, 40 (1992), pp. 642–651.

[2] W. E. Boyse and A. A. Seidl, *A hybrid finite element method for near bodies of revolution*, IEEE Trans. Magnetics, 27 (1991), pp. 3833–3840.

[3] ———, *Convergence acceleration by preconditioning a block quasi minimum residual method for a finite element formulation of electromagnetic scattering*, in URSI Symposium Digest, Chicago, Illinois, 1992, IEEE-APS/URSI/NEM Joint Symposium, p. 172.

[4] ———, *An iterative solver for multiple right hand sides based on the block Lanczos algorithm*, in URSI Symposium Digest, Chicago, Illinois, 1992, IEEE-APS/URSI/NEM Joint Symposium, p. 173.

[5] ———, *A hybrid finite element method for* 3-D *scattering using nodal and edge elements*, IEEE Trans. Antennas Propagation, 42 (1994), pp. 1436–1442.

[6] A. Chatterjee, J. M. Jin, and J. L. Volakis, *Edge-based finite elements and vector ABC's applied to* 3-D *scattering*, IEEE Trans. Antennas Propagation, 41 (1993), pp. 221–226.

[7] J. D'Angelo and I. D. Mayergoyz, *RF scattering and radiation by using a decoupled Helmholtz equation approach*, IEEE Trans. Magnetics, 29 (1993), pp. 2040–2042.

[8] R. W. Freund, *Conjugate gradient type methods for linear systems with complex symmetric coefficient matrices*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 425–448.

[9] R. W. Freund, M. H. Gutknecht, and N. M. Nachtigal, *An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices, Part* I, Tech. report 90.45, RIACS, NASA Ames Research Center, November 1990.

[10] A. George and J. W. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice–Hall, Inc., Englewood Cliffs, NJ, 1981.

[11] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, 1983.

[12] R. F. Harrington, *Field Computation by Moment Methods*, Macmillan, New York, 1968.

[13] J. R. Mautz and R. F. Harrington, *H-field, E-field, and combined-field solutions for conducting bodies of revolution*, Arch. Elektron. Ubertragungstech. (Electron. Comm.), 32 (1978), pp. 159–164.

[14] D. P. O'Leary, *The block conjugate gradient algorithm and related methods*, Linear Algebra Appl., 29 (1980), pp. 293–322.

[15] ———, *Parallel implementation of the block conjugate gradient algorithm*, Parallel Comput., 5 (1987), pp. 127–139.

[16] S. Pissanetzky, *Sparse Matrix Technology*, Academic Press, Inc., New York, 1984.

[17] Y. Saad, *ILUT: A dual threshold incomplete LU factorization*, Res. report UMSI 92/38, University of Minnesota Supercomputer Institute, Minneapolis, MN, March 1992.

[18] D. P. Young, R. G. Melvin, F. T. Johnson, J. E. Bussoletti, L. B. Wigton, and S. S. Samant, *Application of sparse matrix solvers as effective preconditioners*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 1186–1199.

SIAM J. SCI. COMPUT.
Vol. 17, No. 1, pp. 275–286, January 1996

019

SOLVING LINEAR INEQUALITIES IN A LEAST SQUARES SENSE*

R. BRAMLEY† AND B. WINNICKA†

Abstract. In 1980, S.-P. Han [*Least-Squares Solution of Linear Inequalities*, Tech. Report TR–2141, Mathematics Research Center, University of Wisconsin–Madison, 1980] described a finitely terminating algorithm for solving a system $Ax \leq b$ of linear inequalities in a least squares sense. The algorithm uses a singular value decomposition of a submatrix of A on each iteration, making it impractical for all but the smallest problems. This paper shows that a modification of Han's algorithm allows the iterates to be computed using QR factorization with column pivoting, which significantly reduces the computational cost and allows efficient updating/downdating techniques to be used. The effectiveness of this modification is demonstrated, implementation details are given, and the behaviour of the algorithm discussed. Theoretical and numerical results are shown from the application of the algorithm to linear separability problems.

Key words. iterative methods, linear inequalities, least squares, linear separability

AMS subject classifications. 65F10, 65F20, 65F30, 65K05

1. Introduction.

Let $A \in \Re^{m \times n}$ be an arbitrary real matrix, and let $b \in \Re^m$ be a given vector. A familiar problem in computational linear algebra is to solve the system $Ax = b$ in a least squares sense, that is, to find an x^* minimizing $||Ax - b||$. Here and throughout this paper, $\|\cdot\|$ refers to the two-norm. Such an x^* solves the *normal equations* $A^T(Ax - b) = 0$, and the optimal residual $r^* = b - Ax^*$ is unique (although x^* need not be). The least squares problem is usually interpreted as corresponding to multiple observations, represented by the rows of A and b, on a vector of data x. The observations may be inconsistent, and in this case a solution is sought that minimizes the norm of the residuals. More information about linear least squares problems and solution techniques can be found in [9], [15], [4].

A less familiar problem to numerical linear algebraists is to solve systems of linear *inequalities* $Ax \leq b$ in a least squares sense, but the motivation is similar: if a set of observations places upper or lower bounds on linear combinations of variables, we want to find x^* minimizing $||(Ax - b)_+||$, where the ith component of the vector v_+ is the maximum of zero and the ith component of v. However, potential applications extend beyond simple data analysis, and include linear separability problems. That application requires finding a hyperplane that best separates two point sets; when the two sets are not linearly separable, a hyperplane that correctly separates the largest number of points is desired. Although an $\mathcal{L}_1$-norm formulation using linear programming seems more natural for this problem, the $\mathcal{L}_2$-norm formulation described here provides comparable solutions.

When the system $Ax \leq b$ is consistent, that is, when a solution exists that satisfies all the inequalities, then phase I of any standard linear programming method can find it. Furthermore when the system is not consistent, linear programming can identify that case, but does not directly provide an "optimal" solution. Other methods developed for solving linear inequalities include an unusual algorithm by Stewart [14], which defines a function that diverges in a direction that converges to a solution of the inequalities; if no solution exists, the function converges to a unique minimum.

One way of solving the problem is to state it as the quadratic programming problem in (x, z)

*Received by the editors May 23, 1994; accepted for publication (in revised form) December 19, 1994. This work was supported by NSF grant CDA–9223008.

†Computer Science Department, Indiana University, 215 Lindley Hall, Bloomington, IN 47405-4101 (bramley@cs.indiana.edu).

$$(1) \qquad (\mathrm{QP}) = \begin{cases} \min \frac{1}{2} z^T z, \\ \text{subject to } Ax - b \leq z. \end{cases}$$

However, there are serious numerical difficulties with solving a quadratic programming problem that has a singular objective function; furthermore, most methods require an active-set strategy that can be difficult to implement, particularly when it is necessary to decide which entries to drop from the active set. The analogue of an active set for the algorithm described in this paper is automatically determined without difficult decisions of when to drop a constraint. In particular, the *numerical* determination of the active set consists of a test against zero, without the need to introduce machine- or problem-dependent tolerances.

The only algorithm specifically designed for solving arbitrary systems of linear inequalities in a least squares sense was developed by S.-P. Han [6]. That algorithm requires finding the minimum-norm least squares (equality) solution to systems $A_I x = b_I$, where A_I is a submatrix of A consisting of some rows of A. This implies that a singular-value decomposition or a complete orthogonal decomposition of A_I is required on every iteration. Both of these decompositions are relatively expensive to compute, and there are currently no effective update/downdate methods that allow the reuse of work performed on a previous iteration. This paper will show that a minor modification of Han's algorithm allows an implementation using a QR factorization with column pivoting instead, and both the robustness and finite termination of Han's algorithm are retained.

Section 2 of this paper defines notation and reviews some basic properties of least squares solutions for linear inequalities, most of which can be found in [6]. Section 3 outlines Han's algorithm and §4 presents and validates the minor change in the convergence proof that allows QR with column pivoting to be used. Section 5 gives implementation details and testing results. As an illustration of how the algorithm can be used in an application area, §§6 and 7 examine the linear separability problem. Linear separability problems start with two sets of experimental data points, where the points in one set have a certain property and the points in the other do not. A hyperplane that best separates the two point sets is found and then used to classify future data points as having or not having the relevant property.

2. Basics of systems of linear inequalities. This section summarizes some fundamental properties of linear inequalities from Han's technical report, and proofs of the results can be found in [6]. Let $A \in \Re^{m \times n}$ be an arbitrary real matrix, and let $b \in \Re^m$ be a given vector. No relation is assumed between m and n, and the matrix A can be rank deficient, ill conditioned, or even the zero matrix. Let the rows of A be denoted a_i^T, $i = 1, \ldots, m$. We want to find an $x \in \Re^n$ solving the system

$$(2) \qquad Ax \leq b$$

in some sense. System (2) is interpreted componentwise, so we want $a_i^T x \leq b_i$ for all $i = 1, \ldots, m$. Possibly no such x exists. As an example, consider the system

$$A = \begin{bmatrix} 1 & 0 \\ -1 & 0 \end{bmatrix}, b = \begin{pmatrix} 1 \\ -2 \end{pmatrix},$$

which is equivalent to $x_1 \leq 1$ and $x_1 \geq 2$. In these cases, we want a *least squares* solution, which we now define. Given a vector $v \in \Re^l$, define its positive part as the vector $v_+ \in \Re^l$ with components given by $v_+^i = \max\{0, v^i\}$. A least squares solution to (2) is any vector x that minimizes

$$(3) \qquad f(x) = \frac{1}{2} \|(Ax - b)_+\|^2.$$

Analogous to the linear equality case, we can also define the (necessarily unique) x of minimum norm that solves $x = \text{argmin } \|(Ax - b)_+\|$, but the method analyzed here does not provide a minimum-norm solution. For example when $m < n$ and A is full rank, the method finds one of the infinite number of possible solutions, but which one depends on the starting point. Also, the function $f(x)$ is convex, continuously differentiable, and piecewise quadratic. Differentiating gives the analogue of the normal equations.

PROPOSITION 2.1. *$x^* \in \Re^n$ solves* (2) *if and only if $A^T(Ax^* - b)_+ = 0$.*

The proposition follows immediately from the convexity of f and the relation $\nabla f(x) = A^T(Ax - b)_+$.

Keeping in mind the similarity of Proposition 2.1 and the normal equations for equality linear least squares problems, we define the residual vector as $z = (Ax - b)_+$. This residual is zero if and only if the system of inequalities is consistent and x is a solution. Furthermore, by noting the equivalence of (2) and the quadratic programming problem (1), which is convex, we have the following proposition.

PROPOSITION 2.2. *For any matrix $A \in \Re^{m \times n}$ and vector $b \in \Re^n$, a least squares solution to* (2) *exists. The optimal residual vector $z^* = (Ax^* - b)_+$ is unique, and x is a least squares solution if and only if $(Ax - b)_+ = z^*$.*

Finally, we note that the gradient of f is globally Lipschitz of order 1, with a Lipschitz constant of $\|A\|^2$.

PROPOSITION 2.3. $\|\nabla f(x) - \nabla f(y)\| \leq \|A\|^2 \|x - y\|$, *for all $x, y \in \Re^n$.*

Again the proof is straightforward, using the relation $\nabla f(x) = A^T(Ax^* - b)_+$. Note, however, that the Hessian of f fails to exist at points where $a_i^T x = b_i$ for any $i = 1, 2, \ldots, m$.

3. Outline of Han's algorithm. Two notations are needed for the statement of Han's algorithm. First, $G^\dagger$ denotes the pseudoinverse of the matrix G [13]. In practice, all that is needed is the action of $G^\dagger$ on a vector f, not the linear operator itself in explicit form. The vector $G^\dagger f$ is the minimum-norm, least squares solution to the problem of minimizing $||Gy - f||$, and can be computed by a QR factorization when G is full rank, or by singular value or complete orthogonal decomposition otherwise. See [4, Chap. 5] for details on computing these and other factorizations for linear least squares.

Second, let $I \subseteq \{1, 2, \ldots, m\}$ be an index set. Then A_I is the submatrix of A consisting of rows with indices in I. With this definition, $A_I \in \Re^{|I| \times n}$, where $|I|$ is the cardinality of I. The vector b_I can be defined similarly. Using this notation, the algorithm follows.

ALGORITHM 3.1. Let $A \in \Re^{m \times n}$, $b \in \Re^m$, a starting point $x^0 \in \Re^n$, and $\epsilon > 0$ be given.

```
Initialize:
    • Set $k = 0$ (Iteration number)
    • Set $r^0 = b - Ax^0$
    • Set $I = \{i : a_i^T x^0 \geq b_i\} = \{i : r_i^0 \leq 0\}$ (Set of active indices)
    • Set $\rho^0 = ||r_I^0||$ (initial residual norm)
Iterate: While ($\rho^k > \epsilon$)
    • $d^k = A_I^\dagger r_I^k$
    • $\lambda = \text{argmin } f(x^k + \lambda d^k)$, where $f(x)$ is defined by equation (3)
    • $x^{k+1} = x^k + \lambda d^k$
    • $r^{k+1} = b - Ax^{k+1}$
    • $I = I_{k+1} = \{i : a_i^T x^{k+1} \geq b_i\}$ (New set of active indices)
    • $\rho^{k+1} = ||(r^{k+1})_I||$
    • $k = k + 1$
```

Superscripts in the above algorithm indicate the iteration number. Note that I and λ depend on k also, but for clarity, we omit the k when examining a single step of the algorithm. The exact line search for λ is computationally reasonable, since $\theta(\lambda) = f(x^k + \lambda d^k)$ is piecewise quadratic, convex, and continuous; we can simply search through the knot points to isolate an interval on which $\theta(\lambda)$ is quadratic, then interpolate. Numerical testing shows that the algorithm is in fact sensitive to the line-search procedure, and it is worthwhile to consider other methods.

The fundamental result Han established about this algorithm is that it converges in a finite number of steps to some minimizer of (3). The proof relies on the following properties, which are readily verified.

PROPOSITION 3.1. *For any matrix $A \in \Re^{m\times n}$ and vector $b \in \Re^n$, let $f(x)$ be given by* (3), *$x = x^k$, $d = d^k$, and $I = I(x^k)$. Then*

1. *$\nabla f(x) = -A_I^T A_I d$, so $d^T \nabla f(x) = \|A_I d\|^2$, and d is a descent direction for $f(x)$.*
2. *d is the minimum norm least squares solution to the problem: minimize $\|A_I d - r_I\|$.*

4. A more efficient version of Han's algorithm. To make this algorithm practical, d^k needs to be computed by means of a cheaper factorization of A_I. Han's algorithm uses the pseudoinverse to define d^k in order to limit the growth of the component of x^k in the null space of A_I. As mentioned in the introduction, we now show that this can be relaxed, allowing cheaper computation of d^k. Omitting the superscript k temporarily, let $d_{\rm svd} = A_I^\dagger r_I$. The general least squares solution to $A_I d = r_I$ is given by $d = d_{\rm svd} + (I - A_I^\dagger A_I)y$ for $y \in \Re^n$. Note that $(I - A_I^\dagger A_I)y$ is in the null space of A_I, so that $A_I d = A_I d_{\rm svd}$. This is critical since, with one exception, the proof of finite termination in [6] involves only the quantity $A_I d$, not d. Before addressing that exception, we recall a result of Golub and Pereyra [5].

THEOREM 4.1. *Let G be an $m \times n$ matrix, and let $GP = Q\hat{R}$ be the QR with column-pivoting factorization of G. Partition*

$$\hat{R} = \begin{bmatrix} R_1 & R_2 \\ 0 & 0 \end{bmatrix}, \tag{4}$$

where R_1 is $r \times r$, upper triangular, and invertible, and R_2 is $r \times n-r$. Let $y_{\rm svd}$ be the minimum-norm least squares solution to the problem: minimize $\|Gy - f\|$, let $Q^T f = (c^T, g^T)^T$, where $c \in \Re^r$, and set

$$y = P\begin{pmatrix} R_1^{-1}c \\ 0 \end{pmatrix}. \tag{5}$$

Then $\|y\|^2 \le (1 + \|R_1^{-1}R_2\|^2)\|y_{\rm svd}\|^2$.

The application of this result to the linear inequality algorithm tells us that using QR factorization with column pivoting to compute the search direction d will always provide a d that is bounded by a constant times the norm of $d_{\rm svd}$, where the constant does not depend on x or the iteration number k.

THEOREM 4.2. *Let d be computed in Han's algorithm using QR factorization with column pivoting. Then there is $C_R \ge 0$ such that $\|d\| \le C_R\|d_{\rm svd}\|$ independently of the iteration number k.*

Proof. For a given index set I, let R_1 and R_2 be the factors defined by applying Theorem 4.1 to the problem: minimize $\|A_I d - r_I\|$. Let $C_I = \sqrt{(1 + \|R_1^{-1}R_2\|^2)}$, and let $C_R = \max\{C_I\}$, where the maximum is taken over all possible index sets I. Since the number of such index sets is finite, C_R is well defined. The result follows immediately from the last Theorem. □

We now prove the main convergence result needed.

THEOREM 4.3. *Let the sequences of vectors x^k and d^k be computed using Algorithm 3.1, with d^k computed using QR factorization with column pivoting on A_I. Then either the algorithm stops after a finite number of iterations, or*

$$\lim_{k\to\infty} \nabla f(x^k) = 0. \tag{6}$$

Proof. If the algorithm stops after a finite number of iterations there is nothing further to prove. Suppose it takes an infinite number of iterations. Then for all steps, A_I is a nonempty matrix, since $I = \phi$ if and only if x^k is a solution. Initially we drop the superscripts k, and consider one step of the algorithm. Since ∇f is continuous and globally Lipschitz with a Lipschitz constant of $\|A\|^2$, from the mean value theorem [11, Thm. 8.3.1]

$$\begin{aligned} f(x+\lambda d) - f(x) &= \lambda \int_0^1 d^T \nabla f(x + t\lambda d) dt \\ &= \lambda \int_0^1 d^T [\nabla f(x+t\lambda d) - \nabla f(x)] dt + \lambda d^T \nabla f(x) \\ &\le \lambda \int_0^1 \|d\| \|\nabla f(x+t\lambda d) - \nabla f(x)\| dt + \lambda d^T \nabla f(x) \\ &\le \lambda \|A\|^2 \int_0^1 \lambda \|d\|^2 t dt + \lambda d^T \nabla f(x), \end{aligned} \tag{7}$$

and so

$$f(x+\lambda d) \le f(x) + \lambda d^T \nabla f(x) + \frac{\|A\|^2}{2} (\lambda \|d\|)^2. \tag{8}$$

As a function of λ, the right-hand side of the above bound has a minimum at

$$\hat{\lambda} = \frac{-d^T \nabla f(x)}{\|A\|^2 \cdot \|d\|^2}. \tag{9}$$

Substituting this value of λ in (8) and using $d^T \nabla f(x) = -d^T A_I^T A_I d$, we get

$$f(x+\hat{\lambda} d) - f(x) \le -\frac{1}{2} \left[\frac{d^T \nabla f(x)}{\|A\| \cdot \|d\|} \right]^2 \le -\frac{1}{2} \left[\frac{\|A_I d\|^2}{\|A\| \cdot \|d\|} \right]^2. \tag{10}$$

Note that if the component of d that lies in the null space of A_I becomes arbitrarily large, the upper bound above can go to zero. However, from Theorem 4.2 $\|d\| \le C_R \|d_{\text{svd}}\|$, where C_R is independent of the iteration number. Since $d_{\text{svd}} \in$ range (A_I^T), $d_{\text{svd}} = A_I^\dagger A_I d$, and so

$$\|d\| \le C_R \|A_I^\dagger A_I d_{\text{svd}}\| \le C_R \|A_I^\dagger\| \cdot \|A_I d_{\text{svd}}\| \le C \|A_I d_{\text{svd}}\|, \tag{11}$$

where $C = \left(\max_I \|A_I^\dagger\|\right) C_R$. Substituting this bound on d into (10) gives

$$f(x+\hat{\lambda} d) - f(x) \le -\frac{1}{2C\|A\|^2} \|A_I d\|^2. \tag{12}$$

Since the stepsize λ is chosen by an exact line search,

$$f(x) - f(x+\lambda d) \ge f(x) - f(x+\hat{\lambda} d) \ge \frac{1}{2C\|A\|^2} \|A_I d\|^2. \tag{13}$$

The last inequality holds for all iterations, and since $f(x^k)$ is monotone decreasing and bounded below by zero,

$$\sum_{k=0}^{\infty}\left[f(x^k) - f(x^k + \lambda d^k)\right] \geq \frac{1}{2C\|A\|^2}\sum_{k=0}^{\infty}\|A_{I^k} d_k\|^2 \tag{14}$$

is a finite sum, so $A_{I^k}d^k \rightarrow 0$ as $k \rightarrow \infty$ and hence $\nabla f(x^k) = -A_{I^k}^T A_{I^k} d^k \rightarrow 0$ as $k \rightarrow \infty$. □

The rest of Han's proofs, which are not reproduced here, still apply to the modified algorithm, since they only rely on $A_I d$, not d. His arguments in particular show that of the two alternatives in Theorem 4.3, the second alternative (equation (6)) cannot occur. It is worthwhile to compare those results with related ones. As early as 1965, Katznelson [8] established finite termination of an algorithm for solving piecewise linear systems of equations that arise in circuits. More recently Li and Swetits [10] have established finite termination for solving systems of the form $\Phi(x) = Q^T[(Ax+b)_+ + (Cx+d)] = 0$ (c.f. the gradient of $f(x)$). In both cases, they relied on the assumption that within each polyhedral set created by the hyperplanes $H_i = \{x : a_i^T x = b_i\}$, the gradient of Φ is nonsingular. This corresponds to the case where $A_I^T A_I$ is nonsingular for all index sets I. The key idea is that within each polyhedral set the function is quadratic, and so if the kth iterate lands in the polyhedral set containing the (necessarily unique) minimum, Newton's iteration converges in one step. Since there are a finite number of such polyhedral sets, it is a matter of showing that if an infinite number of the iterates lie in a single polyhedral set, they must converge to a point in the set. The unicity of solutions to $\nabla\Phi(x)d = -\Phi$ allows this by assuring that the iterates remain bounded.

Han's method applies a Gauss–Newton approach to the same problem, and restricts the choice of search directions d^k to minimum-norm solutions, in order to have zero growth in the null-space component of A_{I^k}. The key idea used in this paper is that some growth in that null space is allowed, provided that it is uniformly bounded over all of the polyhedral sets, that is, over all choices of index set I. This result is important because it allows applying the algorithm to large systems, and the recent development of efficient and reliable orthogonal factorization methods for sparse systems allows it to be applied to the kind of systems that frequently arise in applications. Furthermore, methods for computing QR factorization with some form of pivoting on parallel machines allow implementation on modern high-performance computers.

5. Numerical characteristics of the algorithm. This section summarizes results of numerical testing, and further details can be found in [2]. The algorithm has been implemented in Matlab©,[1] with four ways of generating search directions: the singular value decomposition (SVD), a complete orthogonal factorization (COF), QR factorization with column pivoting (QR), and QR factorization with updating and downdating of the factors (UD), which is used only when it is numerically safe.

The implementation defines the active set exactly as given in Algorithm 3.1. By contrast, active-set strategies in quadratic programming methods for (1) involve tolerances and can require numerical determination of ranks of submatrices.

The number of iterations required is sensitive to the line search, since stopping short or overshooting a hyperplane boundary gives a different index set. Two line-search methods have been tested: a search through the knot points for the piecewise quadratic line-search function followed by quadratic interpolation, and a binary search method.

Table 1 shows the mean ratios of flops and iterations for 200 random problems of orders 80 × 40, 40 × 80, and 400 × 15. The flops computed are the total flops required, not just those

[1] Matlab© is a registered trademark of The MathWorks, Inc.

TABLE 1

Statistics for comparing costs of SVD and QR for search direction and quadratic interpolation and binary search for line search.

Prob Size	Statistic	$\frac{\text{SVD}}{\text{QR}}$ Flops	$\frac{\text{SVD}}{\text{QR}}$ Iters	$\frac{\text{QuadInt}}{\text{BinSearch}}$ Flops	$\frac{\text{QuadInt}}{\text{BinSearch}}$ Iters
40 × 80	Arith Mean	4.45	0.73	0.98	0.99
	Std Dev	1.82	0.25	0.13	0.12
	Geom Mean	4.13	0.69	0.97	0.99
80 × 40	Arith Mean	6.18	0.95	1.03	1.12
	Std Dev	1.74	0.23	0.26	0.33
	Geom Mean	5.92	0.93	1.00	1.08
400 × 15	Arith Mean	2.04	1.00	1.00	1.01
	Std Dev	0.26	0.00	0.06	0.89
	Geom Mean	2.02	1.00	1.00	1.00

in finding the search direction or performing the line search. The third and fourth columns compare using an SVD to using QR with column pivoting, and show that for all problem sizes, SVD is significantly more expensive, even though fewer iterations are required by SVD, especially for the 40 × 80 problem size. This makes intuitive sense because, in this case, many of the submatrices A_I don't have full column rank. Finding the search direction via QR with column pivoting can introduce components from the null space of A_I, components that may have to be removed by later iterations when the index set changes, and which can introduce numerical instability. However, even when requiring 3 times more iterations, the QR-based method takes fewer flops; less than one-tenth as many in some cases. The fifth and sixth columns show the ratios when QR with column pivoting is used, but this time comparing the two line-search methods. This shows that the two methods are close in cost, but for overdetermined problems binary search is slightly cheaper. Because binary search is easily coded and evaluating the line-search function at a given point requires only the computationally inexpensive task of forming matrix–vector products with A, this suggests it is the preferred method.

Table 2 compares the number of flops for four search direction methods: COF, SVD, QR, and UD. The ratios of the first three to the last one are given, since UD was found most often to be the most efficient method for overdetermined problems. Although the average cost of COF for underdetermined problems is slightly more than for UD, the geometric mean of 0.94 shows that COF is cheaper for a larger number of problems. In this case the subproblems tend to have a large null space for A_I, and eliminating that component as COF does significantly reduces the number of iterations. So although on a given iteration COF is more expensive, its overall cost can be reduced.

The conclusions from these experiments are that using a binary line-search method is cheaper than a quadratic interpolation as often as not, and altering the algorithm to select the search direction with a QR factorization is an important improvement for the $m \geq n$ case. When $n \gg m$ and the extra freedom introduced in null(A_I) could potentially cause numerical difficulties, COF is preferred.

Figure 1 shows the maximum iterations needed for random problems with varying m and n. In all cases no more than $1 + \max(m, n)$ iterations were needed, and the largest number of iterations occurs when $m \approx 2n$, which may be from having twice as many halfspaces as variables, and so finding the correct index set takes several steps. When $m \gg 2n$, few iterations are needed, since there are generally enough active rows to give an impetus in the directions imposed by rows not currently active, but which are active at the solution. Finally,

TABLE 2
Statistics for ratio of flops for search direction methods.

Prob Size	Statistic	$\frac{\text{COF}}{\text{UD}}$	$\frac{\text{SVD}}{\text{UD}}$	$\frac{\text{QR}}{\text{UD}}$
40 × 80	Arith Mean	1.03	4.56	1.00
	Std Dev	0.41	1.81	0.00
	Geom Mean	0.94	4.20	1.00
80 × 40	Arith Mean	1.37	6.20	1.37
	Std Dev	0.38	1.77	0.25
	Geom Mean	1.32	5.95	1.35

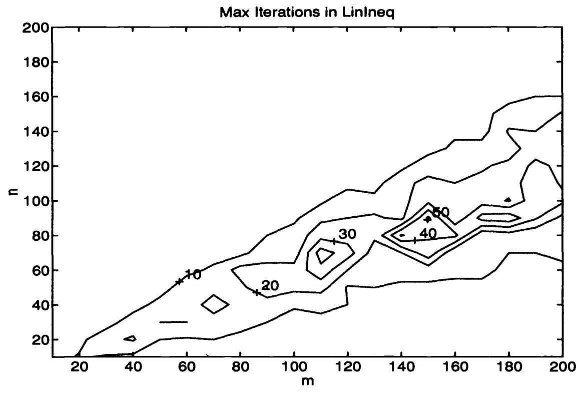


FIG. 1. *Maximum iterations required for m × n problems.*

when $m < n$, only 1–3 iterations are required, since the number of degrees of freedom exceeds the number of "constraints" imposed by the system.

6. Application to the linear separability problem. The linear separability problem is the one of finding a best hyperplane that separates two point sets $\mathcal{A}$ and $\mathcal{B}$ in $\Re^n$. Let $\mathcal{A}$ and $\mathcal{B}$ have m and k points, respectively, and let A and B be matrices with rows giving the coordinates of the points in $\mathcal{A}$ and $\mathcal{B}$. We want to find $w \in \Re^n$ and a scalar γ so that $Aw \leq \gamma e_m$ and $Bw > \gamma e_k$, where $e_i \in \Re^i$ is a vector of all ones.

Clearly not all sets $\mathcal{A}$ and $\mathcal{B}$ can be separated by a hyperplane, so we want to find a hyperplane that is optimal by having fewest points incorrectly classified as belonging to $\mathcal{A}$ or $\mathcal{B}$. This can be approximated as the least squares inequality problem

$$\begin{aligned} Aw - \gamma e_m &\leq -e_m, \\ -Bw + \gamma e_k &\leq -e_k. \end{aligned} \tag{15}$$

This formulation specifies that all the points lie outside of the "slab" $H = \{x : \gamma - 1 \leq w^T x \leq \gamma + 1\}$, guarding against the case when all the data points line up on a hyperplane, in which case the least squares problem (15) with zero on the right-hand side gives a trivial solution $(w, \gamma) = 0$. In [1], Bennett and Mangasarian formulate this problem in terms of the $\mathcal{L}_1$-norm: minimize

$$\frac{1}{m}\sum_{i=1}^{m}\left(-a_i^T w + \gamma + 1\right)_+ + \frac{1}{k}\sum_{j=1}^{k}\left(b_j^T w - \gamma + 1\right)_+ \tag{16}$$

and then solve it as the linear programming problem: minimize over (w, γ, y, z) the function $(y^T e_m)/m + (z^T e_k)/k$ subject to the constraints $Aw - \gamma e_m + y \geq e_m$, $-Bw + \gamma e_k + z \geq e_k$, and $y, z \geq 0$.

The normalization of terms of the objective function by $1/m$ and $1/k$ assures that nontrivial solutions (w, γ) exist. A similar result holds for the least squares formulation (15).

THEOREM 6.1. *Suppose the two point sets $\mathcal{A}$ and $\mathcal{B}$ have m and k points, respectively, and $m, k > 0$. Then the least squares formulation of* (15) *has the trivial solution $w = 0$ if and only if*

$$\sum_{i=1}^{m} a_i = \alpha \sum_{j=1}^{k} b_j \tag{17}$$

for some constant α.

Proof. Suppose that the least squares solution of (15) has $w = 0$, let $\tilde{w} = (w^T, \gamma)^T$, and let

$$G = \begin{bmatrix} A & -e_m \\ -B & e_k \end{bmatrix}, \quad g = \begin{pmatrix} -e_m \\ -e_k \end{pmatrix}. \tag{18}$$

Then

$$\begin{aligned} \|(G\tilde{w} - g)_+\|^2 &= \left\| \begin{pmatrix} (1-\gamma)e_m \\ (1+\gamma)e_k \end{pmatrix}_+ \right\|^2 \\ &= \begin{cases} m(1-\gamma)^2, & \text{if } \gamma < -1, \\ k(1+\gamma)^2, & \text{if } \gamma > 1, \\ m(1-\gamma)^2 + k(1+\gamma)^2, & \text{if } -1 \leq \gamma \leq 1. \end{cases} \end{aligned} \tag{19}$$

Furthermore, the active index set I is

$$I = \begin{cases} \{1, 2, \ldots, m\}, & \gamma < -1, \\ \{m+1, m+2, \ldots, m+k\}, & \gamma > 1, \\ \{1, 2, \ldots, m+k\}, & -1 \leq \gamma \leq 1. \end{cases} \tag{20}$$

From the normal equations, $G^T(G\tilde{w} - g)_+ = 0$. If $\gamma < -1$, this implies

$$0 = \begin{pmatrix} A^T(1-\gamma)e_m \\ -m(1-\gamma) \end{pmatrix} = (1-\gamma)\begin{pmatrix} \sum_{i=1}^m a_i \\ -m \end{pmatrix} \tag{21}$$

and so $m = 0$, a contradiction. Similarly, if $\gamma > 1$, this implies

$$0 = \begin{pmatrix} -B^T(1+\gamma)e_k \\ k(1+\gamma) \end{pmatrix} = (1+\gamma)\begin{pmatrix} -\sum_{j=1}^k b_j \\ k \end{pmatrix} \tag{22}$$

and so $k = 0$, again a contradiction. So $-1 \leq \gamma \leq 1$, in which case

$$0 = \begin{pmatrix} A^T(1-\gamma)e_m - B^T(1+\gamma)e_k \\ -m(1-\gamma) + k(1+\gamma) \end{pmatrix} = \begin{pmatrix} (1-\gamma)\sum_{i=1}^m a_i - (1+\gamma)\sum_{j=1}^k b_j \\ \gamma(m+k) + (k-m) \end{pmatrix}.$$

If $|\gamma| = 1$, the last component implies that one of m or k is zero. So $|\gamma| < 1$, and the theorem holds with $\alpha = (1+\gamma)/(1-\gamma)$. □

It is easy to check condition (17) before beginning computations, and perturbing any entry of G will avoid the trivial solution. By contrast, in the $\mathcal{L}_1$-norm minimizing method of [1], only the objective function for the linear program needs scaling. However, that formulation only guarantees that nontrivial solutions exist, but it does not assure that a linear programming program will find such a nontrivial solution. In any case, the condition in (17) has so far only occurred in artificially created problems.

In testing the linear programming and least squares methods, after (w, γ) are found a secondary minimization on γ is performed to improve the solution. The next proposition shows that the inequality least squares problem will translate the separating hyperplane to where the sum of residual violations is balanced; this is not a criterion of the original linear separability problem, and is the reason we follow the iterations with the secondary minimization.

PROPOSITION 6.2. *Let (w, γ) be a least squares solution to the system* (15). *The sum of residuals corresponding to point set $\mathcal{A}$ equals the sum of residuals corresponding to point set $\mathcal{B}$.*

Proof. From the normal equations for (15),

$$\begin{bmatrix} A^T & -B^T \\ -e_m^T & e_k^T \end{bmatrix} \begin{pmatrix} Aw - (\gamma - 1)e_m \\ -Bw + (\gamma + 1)e_k \end{pmatrix}_+ = 0. \tag{23}$$

The last scalar equation from above gives

$$e_m^T(Aw - (\gamma - 1)e_m)_+ = e_k^T(-Bw + (\gamma + 1)e_k)_+ \tag{24}$$

or equivalently

$$\sum_{i=1}^{m} \left[a_i^T w - (\gamma - 1)\right]_+ = \sum_{j=1}^{k} \left[-b_j^T w + (\gamma + 1)\right]_+ , \tag{25}$$

which is the statement of the proposition. □

The one-dimensional minimization is easily carried out by searching through the knot points defined by $\gamma_i = a_i^T w$ and $\gamma_j = b_j^T w$ and adds little to the overall computation costs. This usually improves the solution by a few data points for two-dimensional test problems, but is much less effective for higher-dimensional problems. Intuitively, in the two-dimensional case minimizing on γ varies one-third of the variables, while for a 9- or 13-dimensional problem, only one-tenth or one-fourteenth of the variables are being changed.

7. Performance on test databases. The linear separability methods have been tested on the Wisconsin Breast Cancer and Cleveland Heart Disease Databases [3]. Here the goal is to provide a linear predictor that can be used to distinguish between benign and malignant tumors in the first database, and patients at risk or not at risk of heart attack in the second database.

Both data sets are available from the University of California–Irvine Repository of Machine Learning Databases and Domain Theories [11]. The first data set consists of 551 points, 346 from set $\mathcal{A}$ (corresponding to benign tumors) and 205 from set $\mathcal{B}$ (corresponding to malignant tumors). Each data point has 9 components, corresponding to experimental measurements. The second data set consists of 297 points, 137 from set $\mathcal{A}$ (corresponding to a negative diagnosis) and 160 from set $\mathcal{B}$ (corresponding to a positive diagnosis).

We discarded data samples that had missing measurements. As in [1], the data were divided randomly into a training group consisting of two-thirds of the data points, and a testing group consisting of the remaining one-third of the data. The best separating hyperplane was found by applying the inequality least squares solver using the training group of data, and then

TABLE 3
Percent of incorrectly classified points for two databases.

Database	Data Group	Ls Sq w/o γ Min	Ls Sq w γ Min	Lin Progr
Cancer	Training	3.19	2.64	3.01
	Testing	4.24	3.80	2.56
Heart	Training	14.75	13.84	15.23
	Testing	15.76	15.96	16.53

the effectiveness of the hyperplane was tested on the remaining testing group of data. This was repeated ten times, with different partitionings into training and testing sets. For the first data set, the inequality solver required seven or eight iterations each time and spent an average of 92% of its flops in finding the search direction. For the second data set, five or six iterations were needed each time and an average of 95% of the flops were spent in finding the search direction.

Table 3 shows the results both with and without the secondary minimization on γ, along with similar results from [1]. The results are not strictly comparable since that work used 566 data points from the Wisconsin Cancer Database and 197 data points from the Cleveland Heart Disease Database, but the comparison suggests that the inequality least squares method provides a solution similar to that of a linear programming method. This is unexpected because the two-norm solution given by inequality least squares can heavily weight outlying data points while the one-norm solution given by the linear programming method weights outlying data points less. However, these results show that for realistic problems the inequality least squares solution is qualitatively competitive.

8. Summary and future work. A computationally efficient implementation of Han's algorithm for solving linear inequalities in a least squares sense has been presented, and has been shown convergent by minor modifications to his convergence proofs. The effectiveness of this change in the algorithm's implementation has been demonstrated, and it has been tested on randomly generated and linear separability problems. The results indicate that with the new implementation, this linear inequalities solution method is a worthwhile addition to a computational scientist's toolkit.

Current work includes applying this method to the graph-partitioning problem for parallel computing (see [7] for a survey of this problem and solution methods). This works in phases: given a graph corresponding to a physical mesh, first find the "deepest" set of nodes by a breadth-first search from the boundary nodes. Two nodes that are furthest apart in the deepest set are selected as initial $\mathcal{A}$ and $\mathcal{B}$ points, and the sets are grown outwards using a constrained breadth-first search. A separating hyperplane is then found to better define those sets, followed possibly by a few steps of simulated annealing to further improve the partitioning. This avoids finding eigenvalues or singular values of the Laplacian of a sparse matrix, and uses only fast graph algorithms and a relatively low-cost linear inequality solver.

The most interesting work remaining is a proof of convergence within $\max(m, n) + 1$ iterations, or a counterexample. Since QR factorization with column pivoting takes $\mathcal{O}(mn^2)$ work in the dense case, this would provide a polynomial upper bound on the algorithm. A linear programming problem can be stated as a system of inequalities [6], so this algorithm would be another polynomial method for linear programming. Furthermore, the subproblems generated by the inequality least squares algorithm have condition numbers no worse than that of the original data, since at each step a decomposition is performed on a submatrix of A (condition number for least squares problems is defined as the ratio of largest to smallest nonzero singular values). Interior point methods for linear programming have subproblems that become increasingly ill conditioned as the iterates converge. So if the conjectured upper bound on the number of iterations holds, this algorithm provides a numerically stable, polynomial time algorithm for linear programming.

REFERENCES

[1] K. Bennett and O. Mangasarian, *Robust linear programming discrimination of two linearly inseparable sets*, Optimization Methods and Software, 1 (1992), pp. 23–34.

[2] R. Bramley and B. Winnicka, *Solving Linear Inequalities in a Least Squares Sense*, Tech. report 396, Indiana University, Bloomington, IN, 1995.

[3] R. Detrano et al., *International application of a new probability algorithm for the diagnosis of coronary artery disease*, American Journal of Cardiology, 64 (1989), pp. 304–310.

[4] G. Golub and C. V. Loan, *Matrix Computations*, 2nd ed., John Hopkins University Press, Baltimore, 1989.

[5] G. Golub and V. Pereyra, *Differentiation of pseudoinverse, separable nonlinear least squares problems and other tales*, in Generalized Inverses and Applications, M. Nashed, ed., Academic Press, New York, 1976, pp. 303–323.

[6] S.-P. Han, *Least-squares solution of linear inequalities*, Tech. Report TR–2141, Mathematics Research Center, University of Wisconsin–Madison, 1980.

[7] B. Hendrickson and R. Leland, *The Chaco user's guide*, Tech. report, Sandia National Laboratory, Albuquerque, NM, 1994.

[8] S. Katznelson, *An algorithm for solving nonlinear resistor networks*, Bell System Technical Journal, 44 (1965), pp. 1605–1620.

[9] C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*, Prentice–Hall, Englewood Cliffs, NJ, 1974.

[10] W. Li and J. Swetits, *A Newton method for solving convex quadratic programs*, SIAM J. Optim., 3 (1993), pp. 466–488.

[11] P. Murphy and D. W. Aha, *UCI Repository of Machine Learning Databases*, http://www.ics.uci.edu/mlearn/MLRepository.html, 1994.

[12] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.

[13] R. Penrose, *A generalized inverse for matrices*, Proc. Cambridge Phil. Soc., 51 (1955), pp. 406–413.

[14] G. Stewart, *An iterative method for solving linear inequalities*, Tech. Report TR–1833, University of Maryland Computer Science Department, College Park, MD, 1987.

[15] G. W. Stewart, *Introduction to Matrix Computations*, Academic Press, New York, 1973.

SIAM J. SCI. COMPUT.
Vol. 17, No. 1, pp. 287–303, January 1996

020

ON THE EFFECTS OF USING THE GRASSMANN–TAKSAR–HEYMAN METHOD IN ITERATIVE AGGREGATION–DISAGGREGATION*

TUĞRUL DAYAR† AND WILLIAM J. STEWART‡

Abstract. Iterative aggregation–disaggregation (IAD) is an effective method for solving finite nearly completely decomposable (NCD) Markov chains. Small perturbations in the transition probabilities of these chains may lead to considerable changes in the stationary probabilities; NCD Markov chains are known to be ill-conditioned. During an IAD step, this undesirable condition is inherited by the coupling matrix and one confronts the problem of finding the stationary probabilities of a stochastic matrix whose diagonal elements are close to 1. In this paper, the effects of using the Grassmann–Taksar–Heyman (GTH) method to solve the coupling matrix formed in the aggregation step are investigated. Then the idea is extended in such a way that the same direct method can be incorporated into the disaggregation step. Finally, the effects of using the GTH method in the IAD algorithm on various examples are demonstrated, and the conditions under which it should be employed are explained.

Key words. Markov chains, decomposability, stationary probability, aggregation–disaggregation, Gaussian elimination, sparsity schemes

AMS subject classifications. 60J10, 60J27, 65F05, 65F10, 60-04

1. Introduction.

NCD Markov chains are irreducible stochastic matrices that can be ordered so that the matrix of transition probabilities has a block structure in which the nonzero elements of the off-diagonal blocks are small compared with those of the diagonal blocks. Such matrices often arise in queueing network analysis, large scale economic modeling, and computer systems performance evaluation. We have

$$P_{n\times n} = \begin{array}{c} \\ \left(\begin{array}{cccc} P_{11} & P_{12} & \cdots & P_{1N} \\ P_{21} & P_{22} & \cdots & P_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ P_{N1} & P_{N2} & \cdots & P_{NN} \end{array}\right) \end{array} \quad \tag{1.1}$$

$$\begin{array}{cccc|c} n_1 & n_2 & \cdots & n_N & \\ \hline P_{11} & P_{12} & \cdots & P_{1N} & n_1 \\ P_{21} & P_{22} & \cdots & P_{2N} & n_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ P_{N1} & P_{N2} & \cdots & P_{NN} & n_N \end{array} .$$

The subblocks P_{ii} are square, of order n_i, with $n = \sum_{i=1}^{N} n_i$. Let the stationary distribution of P, π (i.e., $\pi P = \pi$, $\|\pi\|_1 = 1$), be partitioned conformally with P such that $\pi = (\pi_1, \pi_2, \ldots, \pi_N)$. Each π_i, $i = 1, 2, \ldots, N$ is a row vector having n_i elements. Let $P = \mathrm{diag}(P_{11}, P_{22}, \ldots, P_{NN}) + E$. The quantity $\|E\|_\infty$ is referred to as the *degree of coupling* and it is taken to be a measure of the decomposability of the matrix (see [5]). If it was zero, then P would be reducible.

NCD Markov chains that appear in applications are quite large and sparse, possibly having more than thousands of states. For such large chains, direct methods can generate immense fill-in during the triangularization phase of the solution process, and due to storage limitations they become impractical. Moreover, when the system is nearly completely decomposable, there are eigenvalues close to 1, and the poor separation of the unit eigenvalue implies a slow rate of convergence for standard matrix iterative methods [12]. IAD methods do not suffer from these limitations [7], [23].

*Received by the editors May 18, 1994; accepted for publication (in revised form) November 28, 1994.

†The work of this author was performed while the author was at North Carolina State University, Raleigh, NC. Current address: Computer Engineering and Information Science Department, Bilkent University, Bilkent 06533, Ankara, Turkey (tugrul@bilkent.edu.tr).

‡Department of Computer Science, North Carolina State University, Raleigh, NC 27695-8206 (billy@markov.csc.ncsu.edu).

The idea in IAD methods is to observe the system in isolation in each of the diagonal blocks as if the system is completely decomposable (see [15]) and to compute the stationary probability distribution of each diagonal block. However, there are two problems with this approach. First, since the diagonal blocks are substochastic, the off-diagonal probability mass must somehow be incorporated into the diagonal blocks. Second, the probabilities obtained by this approach are conditional probabilities, and this condition has to be removed by weighing each probability subvector by the probability of being in that group of states. Only if these two problems are overcome can one form the stationary distribution of the Markov chain by weighing the subvectors and pasting them together.

For the transition probability matrix P, the stochastic complement of P_{ii} [8] is

$$S_{ii} = P_{ii} + P_{i*}(I - P_i)^{-1}P_{*i},$$

where

P_{i*} : $n_i \times (n - n_i)$ matrix is composed of the ith row of blocks of P with P_{ii} removed,

P_{*i} : $(n - n_i) \times n_i$ matrix is composed of the ith column of blocks of P with P_{ii} removed,

P_i : $(n - n_i) \times (n - n_i)$ is the principal submatrix of P with ith row and ith column of blocks removed.

Each stochastic complement is the stochastic transition probability matrix of a smaller irreducible Markov chain obtained by observing the original process in the corresponding block of states. The conditional stationary probability vector of the ith block is $\pi_i/\|\pi_i\|_1$ and it may be computed by solving $(\pi_i/\|\pi_i\|_1)S_{ii} = \pi_i/\|\pi_i\|_1$ (see [8] for details). However, each stochastic complement has an embedded matrix inversion that may require excessive computation. An alternative solution technique is to approximate S_{ii} by accumulating the off-diagonal mass P_{i*} into the diagonal block P_{ii} on a row-by-row basis. There are various ways in which this can be done [21]. Thereafter, an approximation to the conditional stationary vector of the block of states may be found by solving the corresponding linear system as before.

To determine the probability of being in a certain block of states, one needs to construct the so-called coupling matrix that shrinks each block down to a single element, forming an $N \times N$ irreducible stochastic matrix. This is accomplished by first replacing each row of each block by the sum of elements in that block row. The sum of elements of row k of P_{ij} gives the probability of leaving state k of block i and entering into block j. Therefore, the operation to be performed for each block is $P_{ij}e$. In what follows, e is a column vector of 1's whose length is determined by the context in which it is used. Moreover, each column vector $P_{ij}e$ must be reduced to a scalar. The total probability of leaving block i to enter into block j may be determined by summing the elements of $P_{ij}e$ after each element of this vector has been multiplied by the probability of being in that state (given that the system is in block i). These multiplicative constants may be obtained from the stationary vector elements; they are the components of $\pi_i/\|\pi_i\|_1$. Hence, the ijth element of the coupling matrix is given by $c_{ij} = (\pi_i/\|\pi_i\|_1)P_{ij}e$. The stationary vector of the coupling matrix gives the stationary probability of being in each one of the block of states. In other words, the weighing factors mentioned before are the elements of the stationary vector of the coupling matrix. However, the stationary vector itself is needed to form the coupling matrix. Since the aim is to compute the stationary vector, one can approximate the coupling matrix by using an approximate stationary vector and improve on the approximate solution iteratively [21].

Without further ado, the IAD algorithm is provided. Convergence analysis of the algorithm appears in [17]. Additional discussion may be found in [21] and [2].

The generic iterative aggregation–disaggregation (IAD) algorithm.

1. Let $\pi^{(0)} = (\pi_1^{(0)}, \pi_2^{(0)}, \ldots, \pi_N^{(0)})$ be a given initial approximation to the solution. Set $k = 1$.

2. Construct the coupling matrix $C^{(k-1)}$

$$c_{ij}^{(k-1)} = \frac{\pi_i^{(k-1)}}{\|\pi_i^{(k-1)}\|_1} P_{ij} e.$$

3. Solve the eigenvector problem

$$\xi^{(k-1)} C^{(k-1)} = \xi^{(k-1)}, \quad \|\xi^{(k-1)}\|_1 = 1$$

for $\xi^{(k-1)} = (\xi_1^{(k-1)}, \xi_2^{(k-1)}, \dots, \xi_N^{(k-1)})$.

4. (a) Compute the row vector

$$z^{(k)} = \left(\xi_1^{(k-1)} \frac{\pi_1^{(k-1)}}{\|\pi_1^{(k-1)}\|_1}, \xi_2^{(k-1)} \frac{\pi_2^{(k-1)}}{\|\pi_2^{(k-1)}\|_1}, \dots, \xi_N^{(k-1)} \frac{\pi_N^{(k-1)}}{\|\pi_N^{(k-1)}\|_1} \right).$$

(b) Solve the N systems of equations

$$\pi_i^{(k)} = \pi_i^{(k)} P_{ii} + \sum_{j>i} z_j^{(k)} P_{ji} + \sum_{j<i} \pi_j^{(k)} P_{ji}$$

for $\pi_i^{(k)}$, $\quad i = 1, 2, \dots, N$.

5. Test $\pi_i^{(k)}$ for convergence. If the desired accuracy is attained, then stop and take $\pi_i^{(k)}$ as the stationary probability vector of P. Else set $k = k + 1$ and go to step 2.

In the IAD algorithm, steps 2 and 3 form the aggregation step and step 4(b), which is essentially a block Gauss–Seidel iteration, forms the disaggregation step. In step 2, $\pi_i^{(k-1)}/\|\pi_i^{(k-1)}\|_1$ is an approximation of the stationary distribution of the stochastic complement of P_{ii}. The weighing factors $(\|\pi_1\|_1, \|\pi_2\|_1, \dots, \|\pi_N\|_1)$ are approximated by $\xi^{(k-1)}$ in step 3. Each iteration of the IAD algorithm reduces the residual error (i.e., $\|\pi(I - P)\|$) by a factor of $\|E\|$ (see [14], p. 80).

The next section discusses how a modified version of Gaussian elimination may be used to enforce stability in the solution of the coupling matrix. In §3, the idea is extended so that it can be used in a nonsingular system of equations with a substochastic coefficient matrix. Section 4 discusses certain implementation issues, and §5 provides numerical experiments with the IAD algorithm on NCD Markov chains.

2. Solving the coupling matrix. The coupling matrix is an irreducible stochastic matrix of order N; that is, each of its two dimensions is equal to the number of blocks in the NCD Markov chain, and all states form a single communicating class. The goal is to solve the singular system $\xi^{(k)} C^{(k)} = \xi^{(k)}$ subject to $\|\xi^{(k)}\|_1 = 1$, which is termed the normalization equation. Essentially, each row of $C^{(k)}$ is a linear combination of the others, and inclusion of the normalization equation enables one to replace the redundant equation, thus achieving full rank in the set of equations. Being an irreducible stochastic matrix, the coupling matrix has a unique unit eigenvalue. All other $N - 1$ eigenvalues are close to 1. The distance of these other eigenvalues to 1 depends on the degree of coupling.

A careful inspection reveals that one can solve the equivalent transposed system

$$(I - C^{(k)})^T (\xi^{(k)})^T = 0, \qquad \|\xi^{(k)}\|_1 = 1. \tag{2.1}$$

The justification for considering this form of the problem is the following. (2.1) is the conventional form in which a (homogeneous) linear system of equations is expressed: the coefficient matrix postmultiplied by the unknown vector equals the (zero) right-hand side. Besides, there are already existing algorithms that may be used with this form of a linear system. But most importantly, as it is explained later in §4, a row-wise sparse storage implementation will spend extra time in the substitution phase of the nontransposed system of equations.

$(I - C^{(k)})^T$ is a singular M-matrix (see [1]) with 0 column sums, and the unique null vector of unit 1-norm is sought. For such a matrix, Gaussian elimination (GE) preserves column diagonal dominance throughout its computation so that the multiplier element at each step is bounded by 1 thereby eliminating the need for pivoting. This follows from the fact that the pivot element at a given step has the largest magnitude among all elements that lie in the unreduced part of its respective column. Note that the same argument would be valid for the original system of equations if $C^{(k)}$ was doubly stochastic.

It is well known that since $C^{(k)}$ is a perturbation of the identity matrix, all its nonunit eigenvalues are close to 1. Due to this fact, iterative methods tend to converge slowly. On the other hand, certain stability issues need to be addressed if direct methods are used. As it is shown next, ordinary GE is not stable in the presence of rounding errors on a coupling matrix whose diagonal elements are close to 1.

Example. Consider the following irreducible stochastic matrix:

$$P = \begin{pmatrix} \frac{1}{3} & \frac{2}{3} - \alpha & \alpha \\ \frac{3}{4} - \alpha & \frac{1}{4} & \alpha \\ \alpha & \alpha & 1 - 2\alpha \end{pmatrix}.$$

Assuming α is small, P is clearly NCD. Additionally, the degree of coupling is 2α. Now, suppose 2α is less than machine epsilon (smallest representable positive floating-point number ϵ such that $1 + \epsilon > 1$). Let $\pi^{(k)} > 0$. Then, the coupling matrix is rounded to

$$C^{(k)} = \begin{pmatrix} 1 & \alpha \\ 2\alpha & 1 \end{pmatrix}.$$

Note that the coupling matrix is not stochastic. Therefore, the matrix $(I - C^{(k)})^T$ is rounded to

$$(I - C^{(k)})^T = \begin{pmatrix} 0 & -2\alpha \\ -\alpha & 0 \end{pmatrix}.$$

Hence, it is necessary to apply some sort of pivoting strategy in order to proceed with GE. Otherwise, the algorithm breaks down contrary to the general belief that GE applied to stochastic matrices is always stable. Before going any further, this issue should be explained some more.

It is possible for an irreducible NCD Markov chain ordered as described in (1.1) to have zero blocks. Consequently, the coupling matrix of such a chain at any iteration has zeros in locations corresponding to zero blocks in the transition probability matrix. However, the zeros in the coupling matrix occur in just the right places so that the coupling matrix is still irreducible [8]. Keeping this in mind, the situation that causes GE to break down occurs when the pivot element at a given step of GE is zero in the matrix $(I - C^{(k)})^T$ during the kth iteration of the IAD algorithm. A sufficient condition for the failure of GE during the aggregation step is to have a transition probability matrix with a degree of coupling less than machine epsilon. In this case, all diagonal elements in $(I - C^{(k)})^T$ will be zero, as in the example, and GE will fail in the first step.

The advocated approach to finding a remedy for this situation is the GTH algorithm described in [4] and the direct method discussed in [16]. The original GTH algorithm emerges from probabilistic arguments, and it is shown that the stationary distribution of a Markov chain can be calculated using only nonnegative numbers and avoiding subtraction operations.

This algorithm achieves significantly greater accuracy than other algorithms described in the literature [11], [5] since there is no loss of significant digits due to cancellation [6]. Interestingly, the inspiration for the algorithm presented in [16], which is specifically for the solution of NCD Markov chains, is the GTH algorithm. In a recent paper [10], it is shown that the stationary probability vector of an $N \times N$ irreducible Markov chain stored in floating-point form is close to its exact stationary vector. In other words, even if one has an algorithm that does not introduce any errors by itself, the best that can be done is to compute the stationary vector of an $N \times N$ Markov chain with an entrywise relative error of only about $2Nu$ of the exact stationary vector. In fact, the relative error incurred by each element of the floating-point stationary vector is about $2Nu$, where u is the unit roundoff (i.e., the maximum relative error in approximating a real number by its nearest floating-point number). This result follows from the fact that the perturbation introduced by storing the matrix in floating-point form is of order unit roundoff. Moreover, in the same paper it is proven that if GTH is employed in the solution process of this $N \times N$ Markov chain, the relative error in each entry of the stationary vector will be of order N^3u. It should be remarked that the entrywise relative error in the stationary vector for the GTH algorithm is independent of the structure of the matrix and the magnitude of its elements.

The following lemma shows that the GTH way of calculating the pivot element may also be applied to the transposed system of equations (2.1). It is this form of the equations in which multipliers are bounded by 1, and it will be demonstrated by numerical experiments that the type of implementation chosen (i.e., transposed versus nontransposed system of equations) has no effect on the accuracy of the results obtained.

LEMMA. Let $A = I - P^T$, where P is a stochastic matrix, and

$$A = \begin{pmatrix} a_{11} & v^T \\ u & A_{22} \end{pmatrix}.$$

Then $e^T A = 0$ and $e^T u = -a_{11}$. If

$$L_1 A = \begin{pmatrix} a_{11} & v^T \\ 0 & \tilde{A}_{22} \end{pmatrix},$$

where

$$L_1 = \begin{pmatrix} 1 & 0 \\ -u/a_{11} & I \end{pmatrix}, \qquad L_1^{-1} = \begin{pmatrix} 1 & 0 \\ u/a_{11} & I \end{pmatrix},$$

then $\tilde{A}_{22}$ (like A) is a singular M-matrix having 0 column sums.

Proof. We have

$$e^T A = e^T (L_1^{-1} L_1) A = 0,$$

$$e^T L_1^{-1} = (1, 1, \dots, 1) \begin{pmatrix} 1 & 0 \\ u/a_{11} & I \end{pmatrix} = (0, 1, \dots, 1),$$

$$(0, 1, \dots, 1)(L_1 A) = 0 \;\Rightarrow\; e^T \tilde{A}_{22} = 0.$$

Other steps may be shown similarly. Note that the lemma is also valid for the negation of any generator matrix. □

The properties of a singular M-matrix coupled with the GTH idea of avoiding subtractions and negative numbers suggests the following modification to GE. At each step of GE, rather than calculating the pivot element in the usual way, one can correct the pivot by replacing it with the negated sum of the off-diagonal elements in the unreduced part of the same column as the pivot. When one mentions the GTH method, it is this approach used in calculating the pivot elements that is implied.

3. Using the GTH method in the disaggregation step. In a given iteration, the disaggregation step of the IAD method uncouples the NCD Markov chain to obtain a new estimate for the stationary distribution. As indicated in [21], in order to achieve an even better approximation of π, at the $(k+1)$st iteration of the IAD algorithm, one solves

$$(I - P_{ii}^T)(\pi_i^{(k+1)})^T = b_i, \tag{3.1}$$

where $\pi_i^{(k+1)}$ is the $(k+1)$st approximation of π_i and $b_i^T = \sum_{j<i} \pi_j^{(k+1)} P_{ji} + \sum_{j>i} z_j^{(k+1)} P_{ji}$ for $i = 1, 2, \ldots, N$ (see step 4(b) of the IAD algorithm).

P_{ii} is a strictly substochastic matrix of order n_i, and $b_i \neq 0$, which gives a nonhomogeneous system of linear equations with a nonsingular coefficient matrix. Had the coefficient matrix been stochastic, the same GTH technique that is utilized in solving the coupling matrix could be clearly employed. In order to be able to use the GTH technique in solving the diagonal block system, first certain modifications must be made. These modifications involve adding one more equation and augmenting the matrix with b_i to put the system into the form

$$W_i^T (\pi_i^{(k+1)}, \hat{\pi}_i^{(k+1)})^T = 0, \tag{3.2}$$

where

$$W_i^T = \begin{array}{c} \begin{array}{cc} n_i & 1 \end{array} \\ \left(\begin{array}{cc} I - P_{ii}^T & b_i \\ w_i^T & \hat{w}_i \end{array} \right) \end{array} \begin{array}{c} \\ n_i \\ 1 \end{array},$$

$w_i = P_{i*}e$, $\hat{w}_i = -b_i^T e$, and $\hat{\pi}_i^{(k+1)}$ is introduced so that the solution vector has as many columns as the coefficient matrix. In other words, $(w_i^T, \hat{w}_i)$ sums up the columns of W_i^T to 0. The values of $\hat{w}_i$ and $\hat{\pi}_i^{(k+1)}$ are irrelevant, because just as in (2.1) one has a singular system with $n_i + 1$ equations, and after W_i^T is reduced to upper-triangular form, the last row will be all 0's. Hence, the reduction needs to be carried out for only n_i steps. What needs to be done for the computation of $(\pi_i^{(k+1)})^T$ is to use the first n_i elements of column $n_i + 1$ in the upper-triangular matrix as the right-hand side in the back substitution phase.

It is not possible to put GE to use in the disaggregation step in the previous example in §2 or in similar problems on the diagonal blocks for which $\|P_{i*}\|_\infty < \epsilon$. Under the given condition, $I - P_{ii}^T$ is a singular matrix in floating-point form; however, the vector b_i is still nonzero, and therefore the system of equations in (3.2) is inconsistent. On the other hand, GTH computes w_i^T in W_i^T by using the nonzero elements in P_{i*} and forms the pivot by summing off-diagonal entries in the unreduced part of the same column as the pivot. Hence, GTH may be applied to solve such blocks.

4. Implementation considerations. As mentioned before, NCD Markov chains that arise in real-life applications are generally large and sparse. This necessitates the design and employment of sparse storage schemes, which essentially store only the nonzero elements of the transition probability/rate matrix.

So far, the application of the GTH (and GE, for that matter) algorithm to systems (2.1) and (3.1) is considered. (3.1) is written in the form of (3.2) so that it has a singular M-matrix with 0 column sums as its coefficient matrix just as in (2.1). Now, the alternative nontransposed systems of equations may be written. The system that corresponds to (2.1) is given by

$$\xi^{(k)}(I - C^{(k)}) = 0, \qquad \|\xi^{(k)}\|_1 = 1. \tag{4.1}$$

Similarly, the equivalent of (3.2) is

$$(\pi_i^{(k+1)}, \hat{\pi}_i^{(k+1)})W_i = 0. \tag{4.2}$$

Define

Scheme 1. The transposed systems of equations in (2.1) and (3.2).

Scheme 2. The nontransposed systems of equations in (4.1) and (4.2).

Unless otherwise specified, reductions mean row reductions (i.e., the addition of a multiple of a lower-indexed row to a higher-indexed row) in a given system of equations. The last assumption is that the coefficient matrices are supplied to both schemes in the nontransposed version. This is a fairly reasonable assumption, since the matrices are usually generated by following the possible transitions from a given state, implying a row-wise generation.

The advantage of reducing the coefficient matrices in Scheme 1 rather than the ones in Scheme 2 to upper-triangular form is twofold:

- the multiplier elements at each step of the reduction are bounded by 1 and
- only the upper-triangular matrix needs to be stored during the reduction process.

Although the multipliers in Scheme 2 are not necessarily bounded by 1, the growth factor still cannot be greater than 1 as indicated in [22]. However, if row reductions are carried out in Scheme 2, both the upper-triangular matrix and the lower-triangular matrix, which contains the multipliers, have to be stored during the triangularization process. The reason is that if row reductions are performed on the nontransposed coefficient matrix, the LU decomposition will provide a nonsingular lower-triangular matrix and a singular upper-triangular matrix that has 0's in the last row. Although the decomposition is premultiplied by the stationary vector, one only has to carry out a single substitution phase (see [9], p. 724) just as in the transposed system of equations (but this time involving the lower-triangular matrix), thereby making it necessary to store both the lower and the upper-triangular matrix during the reduction. One cure that comes to mind is to store only the upper-triangular matrix during the reduction process and to reconstruct the multiplier matrix after the reduction is over. Though quite straightforward, this solution is inefficient due to the superfluous operations performed in calculating the multipliers for the second time. On the other hand, Scheme 1 calls for the transposition of the coefficient matrix before executing GE or GTH.

What follows is a discussion of the effects of the two implementation schemes on a *semisystematic row-wise sparse storage format* (see [13]). A row-wise sparse storage format is a data structure that stores the nonzero elements in the coefficient matrix row by row. Semisystematic means the elements of row i precede those of row $i + 1$, but the elements within a given row need not be ordered. Together with the real value of each nonzero element, the column index of the element is stored as well. Hence, two arrays of the same length, one real and the other integer, are needed. Last, in order to have access to the rows, the starting location of each row has to be stored in an integer array. To facilitate the computation of the number of nonzero elements per row, an extra element whose value is equal to the total number of nonzero elements plus 1 is appended to this array.

The GE algorithm, no matter which scheme is chosen, may be implemented using delayed updates, which means at step k all reductions on row $k + 1$ may be carried out and the row compacted and stored before the algorithm continues in the next step with the update of the next row. The property of the GE algorithm that makes this efficient implementation possible is that reduction on a given row requires the addition of appropriate multiples of lower-indexed rows to it. Hence, there is no data dependency between a row to be reduced and higher-indexed rows in GE. The importance of this idea lies in the fact that it is sufficient to fully expand only the row to be updated at a given step, whereas all other rows with higher indices can still be held in compact form. However, this is no longer applicable when GTH is used in Scheme 1, since all updates due to a row must be finished before the algorithm proceeds with the next step. This situation is dictated by the new way of calculating the pivot elements. In the GTH algorithm, the pivot element is formed by summing the most recently updated elements that lie below the pivot, which implies a dependency of data between the pivot row and the higher-indexed rows. Since storage is limited, one possibility is to expand each row that is operated on, update it, and finally store it back in the appropriate location. The drawback is that this sequence of operations on a row, depending on the fill-in, most probably causes other elements in sparse storage to be shifted around the data structure (extra nonzero entries are likely to occur after a row update). Consequently, it is safe to assume that GTH, with such an implementation, spends more time doing memory reads and writes for larger chains. As pointed out in §2, there is a slight difference in the way the single substitution phase in Schemes 1 and 2 is implemented in the GTH algorithm when a row-wise sparse storage format is used. In the substitution phase, Scheme 2 calls for the solution of a homogeneous linear system in which the stationary vector is postmultiplied by the lower-triangular matrix obtained from the LU decomposition. The substitution is accomplished by accessing a different column of the lower-triangular matrix at each step. This situation makes it necessary to have a doubly nested loop in the code for accessing the elements of the columns. On the other hand, the substitution phase of Scheme 1 requires the solution of a homogeneous linear system in which the stationary vector is premultiplied by an upper-triangular matrix. This being so, the substitution phase of Scheme 1 conforms nicely to a row-wise sparse storage implementation, and, therefore, it may be handled in a single loop that accesses a row of the upper-triangular matrix at each step. In the actual implementation, the normalization equation (i.e., $\|\pi\|_1 = 1$) is used to discard the last equation and form a nonzero right-hand side in both schemes.

A suggestion made by G. W. Stewart [18] as a compromise between the high accuracy of GTH and the implementation efficiency of GE is the following. Rather than performing a correction on the pivot element at each step of GTH, the suggestion is that a pivot should be corrected only when cancellation would produce an inaccurate pivot. For the systems arising in the IAD algorithm, cancellation seems to occur at quite predictable places. Specifically, for the coupling matrix $C^{(k)}$, cancellation occurs in the computation of the diagonal elements of $I - C^{(k)}$. The suggested cure is to compute the diagonal elements of $I - C^{(k)}$ from the off-diagonal elements at the start and to use GE thereafter. For the solution of the diagonal blocks, cancellation is most likely to occur in the computation of the last pivot element. In this case the recommendation is that GE should be applied to the augmented system in (3.2) (or (4.2)), and only the last pivot should be computed by summing the off-diagonal probability mass as in GTH. It is expected that using the suggested scheme in the IAD algorithm will result in a competitive solver (with the GTH method in terms of relative accuracy and with the GE method in terms of total time spent in the IAD algorithm). Results of experiments with this scheme are discussed in §5.

An alternative storage scheme is one that keeps the unused array elements in the form of a linked list (basically, a freepool). In this case, an extra integer array, which, for each nonzero element, stores the location of the next nonzero element in the same row, is required. If a given element is the last in its row, then the corresponding pointer may be set to 0. Though this scheme circumvents the problem of moving array elements back and forth in memory, it introduces overhead due to the manipulation of the freepool. Note that just as in the ordinary row-wise sparse storage format, all elements in a row may be kept ordered according to their column indices in this last implementation. Finally, it should be pointed out that the effect of using a column-wise sparse storage format with Scheme 2 is the same as using a row-wise sparse storage format with Scheme 1 and vice versa.

Although not as costly as the extra reads and writes due to the discussed implementation problems, GTH has $N - k$ extra additions at step k of the elimination procedure, amounting to $\sum_{k=1}^{N-1}(N - k) = N(N - 1)/2$ extra operations in the solution of the coupling matrix. Similarly, there are $n_i(n_i - 1)/2$ extra additions in the solution of the ith diagonal block. Clearly, this is an upper bound. Since these extra addition operations are performed only on nonzero elements in a sparse storage implementation, the actual overhead may be smaller. Moreover, because of the need to introduce one more row to the linear system, extra $n_i + 1$ locations are used during the solution of each P_{ii} in the disaggregation step. Last, in order to form the $(n_i + 1)$st equation during the disaggregation step, a number of additions equal to the number of nonzero elements in P_{i*} are performed. If extra memory locations are available, these N matrix–vector multiplications may be performed at the outset and the resulting vectors stored for future use.

Note that both programming difficulties and overhead storage (arrays to store the indices of nonzero array elements and pointers between elements) increase with the sophistication of the storage scheme. Together with the row-wise sparse storage format, experiments with the sparse storage format proposed by Knuth (see [13] for a brief explanation), which provide equally fast access to columns and rows of a matrix, are conducted. For each nonzero element, in addition to having a pointer to the next nonzero element in the same row, there is an additional pointer to the next nonzero element in the same column. Obviously, there are two arrays that keep the locations of the first nonzero element in each row and each column, respectively. However, as remarked earlier, extra time will be spent for obtaining memory elements from and returning memory elements to the freepool with this kind of implementation. With Knuth's sparse storage format, there is no need to transpose the matrix because rows may be treated as columns and columns as rows. Although one may be inclined to think that Knuth's sparse storage format is more advantageous than the row-wise sparse storage format, due to the overhead of handling the freepool, numerical experiments suggest that Knuth's format is likely be useful only if implemented in a programming language that provides primitives for pointers to memory.

If some information about the structure of the coefficient matrix is available, then another possibility may be to employ a two-dimensional banded storage structure. As for GE, the row-wise sparse storage format with Scheme 1 is recommended.

Keeping in mind that memory is much slower than CPU, one must evaluate the significance of extra operations performed and excess time spent in a sparse storage implementation as opposed to the increased stability and accuracy that accrue from GTH in the context of real-life applications.

5. Numerical results. Experiments with the IAD algorithm are carried out in sparse storage on a SUN SPARC station 2. All routines used are part of the software package MARCA (Markov Chain Analyzer) (see [20]). The routines are written in FORTRAN and compiled in both double-precision and quadruple-precision floating-point arithmetic. For each problem solved, the residual error and the relative error in the solution are computed. The relative error is computed as $\|\pi - \tilde{\pi}\|_2/\|\pi\|_2$, where π is the quadruple-precision solution and $\tilde{\pi}$ is the double-precision solution obtained by the IAD algorithm. Both π and $\tilde{\pi}$ are normalized so that their 1-norms are unity. The residual error is computed as $\|(I - P^T)\tilde{\pi}^T\|_2$, which theoretically is 0 if $\pi = \tilde{\pi}$. See Table 1.

TABLE 1
Notation for parameters of numerical methods.

n	Order of the stochastic transition matrix P
n_z	Number of nonzero elements in the matrix
N	Number of strongly connected components
nzb	Number of nonzero blocks
$Iter$	Number of iterations required to get a residual norm of less than 10^{-15}
T_{total}	Total time spent in the IAD algorithm, (in CPU seconds)
Err_{res}	$\|(I - P^T)\tilde{\pi}^T\|_2$
Err_{rel}	$\|\pi - \tilde{\pi}\|_2/\|\pi\|_2$
γ	Decomposability parameter
M_{agg}	Method used in aggregation phase
M_{disagg}	Method used in disaggregation phase

TABLE 2
Solvers used.

GE_1	Sparse Gaussian elimination using Scheme 1
GE_2	Sparse Gaussian elimination using Scheme 2
GTH_2	Sparse Grassmann–Taksar–Heyman algorithm using Scheme 2
GTH_1	Sparse Grassmann–Taksar–Heyman algorithm using Scheme 1

The problem associated with GE, when NCD Markov chains with a degree of coupling less than machine epsilon are to be solved, has already been shown. Therefore, the emphasis is on problems with a degree of coupling larger than machine epsilon in order to see how GE and GTH behave comparatively in this case. Experiments are performed with implementation Schemes 1 and 2 on the row-wise sparse storage format used in MARCA. The GTH implementation of Scheme 1 is accomplished by shifting contents of arrays rather than using the alternative with the freepool. See Table 2.

The first problem appears quite frequently in the literature. Although the order of the matrix considered in this problem is quite small, it is still instructive to examine the effects of using the IAD algorithm on such a problem. The second problem investigated is a real-life example and had been studied with a variety of parameters in the past. The scheme suggested by G. W. Stewart [18] in §4 is implemented, and the outcome of the related experiments are discussed following the presentation of the numerical results.

The decomposability parameter γ (in Table 1) is input by the user; it is used to determine the strongly connected components in the transition matrix by simply ignoring the elements that are less than the suggested value. Therefore, γ may be taken as an approximation of the degree of coupling. If the matrix is not already in the form (1.1), then symmetric permutations are used to put it into the form in which the diagonal blocks form the strongly connected aggregates.

5.1. Test Problem 1. The first problem that is considered is the 8×8 Courtois matrix [3] with all row sums equal to 1. The degree of coupling for this matrix is 0.001.

TABLE 3
Results for Problem 1: $n = 8, n_z = 41, \gamma = 0.001, N = 3, nzb = 9$.

M_{agg}	M_{disagg}	$Iter$	T_{total}	Err_{res}	Err_{rel}
GE_1	GE_1	4	0.04	$0.286E-16$	$0.824E-13$
GE_2	GE_2	4	0.04	$0.347E-16$	$0.904E-13$
GTH_2	GTH_2	4	0.04	$0.250E-16$	$0.420E-15$
GTH_1	GTH_1	4	0.04	$0.208E-16$	$0.282E-15$

$$P = \begin{pmatrix} 0.85 & 0 & 0.149 & 0.0009 & 0 & 0.00005 & 0 & 0.00005 \\ 0.1 & 0.65 & 0.249 & 0 & 0.0009 & 0.00005 & 0 & 0.00005 \\ 0.1 & 0.8 & 0.0996 & 0.0003 & 0 & 0 & 0.0001 & 0 \\ 0 & 0.0004 & 0 & 0.7 & 0.2995 & 0 & 0.0001 & 0 \\ 0.0005 & 0 & 0.0004 & 0.399 & 0.6 & 0.0001 & 0 & 0 \\ 0 & 0.00005 & 0 & 0 & 0.00005 & 0.6 & 0.2499 & 0.15 \\ 0.00003 & 0 & 0.00003 & 0.00004 & 0 & 0.1 & 0.8 & 0.0999 \\ 0 & 0.00005 & 0 & 0 & 0.00005 & 0.1999 & 0.25 & 0.55 \end{pmatrix},$$

$$\pi^T = \begin{pmatrix} 0.8928265275450187E-01 \\ 0.9275763750513320E-01 \\ 0.4048831201636394E-01 \\ 0.1585331908198259E+00 \\ 0.1189382069041751E+00 \\ 0.1203854811060527E+00 \\ 0.2777952524492734E+00 \\ 0.1018192664446740E+00 \end{pmatrix}.$$

5.2. Test Problem 2. The second problem considered appears in [19]. It represents a time-shared, multiprogrammed, paged, virtual memory computer system modeled as a closed queueing network (see Fig. 1).

The system consists of a number of users using the terminals, a central processing unit (CPU), a secondary memory device (SMD), and a filing device (FD). The degree of multiprogramming in the system at any given time is $\eta = \eta_0 + \eta_1 + \eta_2$, where η_0, η_1, η_2 are, respectively, the number of processes in the CPU, SMD, and FD queues at that moment. Furthermore, η_t represents the number of processes currently being generated at the terminals but not yet transmitted to the CPU for execution. λ is the mean generation time of processes by users working on the terminals. The mean service rate of the CPU depends on η and is given by $\mu_0(\eta)$. The processes that leave the CPU proceed to the SMD, to the FD, and back to the terminals with probabilities p_1, p_2, $1 - p_1 - p_2$, respectively. The mean service rate at the SMD is given by μ_1 and at the FD by μ_2. All arrival and service rates are exponential. Note that it is not possible to apply analytical techniques to solve the above model since the CPU execution time depends on the degree of multiprogramming.

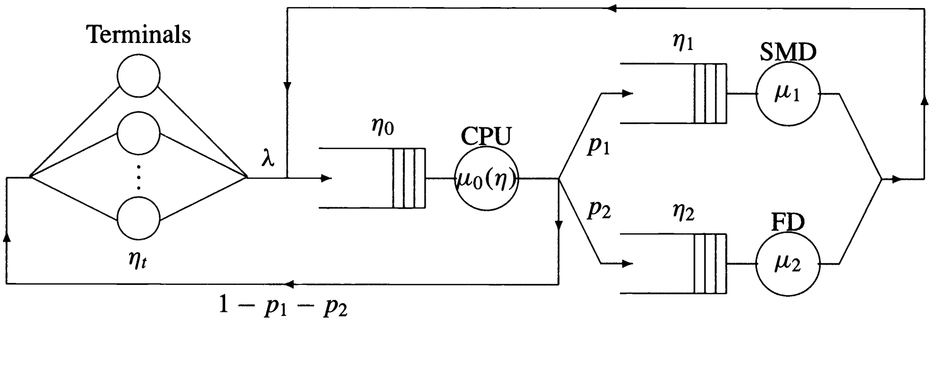


FIG. 1. *An interactive computer system.*

Given the necessary parameters, MARCA generates the transition probability matrix corresponding to the specific queueing system under consideration. Experiments with different combinations of parameters are carried out because the effect of making the transition probability matrix larger and closer to the identity matrix so that it is almost decomposable is to be observed. The generated matrix has a block tridiagonal structure with a small percentage of nonzero elements. It must be remarked that this type of matrix is frequently encountered in queueing network analysis.

(a) For the first example, the following parameters are chosen:

$$\begin{aligned} \eta_t + \eta &= 3, \\ \lambda &= (10^{-4})\eta_t, \\ p_1\mu_0(\eta) &= 100(\eta/128)^{1.5}, \\ p_2\mu_0(\eta) &= 0.05, \\ 1 - p_1 - p_2 &= 0.002, \\ \mu_1 &= 0.2, \\ \mu_2 &= 1/30. \end{aligned}$$

TABLE 4
Sizes of the aggregates for Problem 2(*a*):$\gamma = 10^{-3}$.

10	6	3	1

TABLE 5
Results for Problem 2(a): $n = 20, n_z = 80, \gamma = 10^{-3}, N = 4, nzb = 10$.

M_{agg}	M_{disagg}	*Iter*	T_{total}	Err_{res}	Err_{rel}
GE_1	GE_1	5	0.10	$0.468E-15$	$0.381E-12$
GE_2	GE_2	5	0.10	$0.483E-15$	$0.391E-12$
GTH_2	GTH_2	5	0.10	$0.447E-15$	$0.404E-12$
GTH_1	GTH_1	5	0.10	$0.448E-15$	$0.404E-12$

(b) The parameters in this example are the same as those in 2(a) except

$$\mu_2 = (10^{-10})/30.$$

Sizes of the aggregates for Problem 2(b) for $\gamma = 10^{-11}$ are the same as for Problem 2(a).

TABLE 6
Results for Problem 2(b): $n = 20, n_z = 80, \gamma = 10^{-11}, N = 4, nzb = 10$.

M_{agg}	M_{disagg}	*Iter*	T_{total}	Err_{res}	Err_{rel}
GE_1	GE_1	1	0.07	$0.974E-17$	$0.345E-14$
GE_2	GE_2	1	0.07	$0.974E-17$	$0.345E-14$
GTH_2	GTH_2	1	0.07	$0.999E-20$	$0.421E-16$
GTH_1	GTH_1	1	0.07	$0.999E-20$	$0.421E-16$

(c) The parameters in this example are the same as those in 2(a) except

$$\mu_2 = (10^{-14})/30.$$

Sizes of the aggregates for Problem 2(c) for $\gamma = 10^{-15}$ are the same as for Problem 2(a).

TABLE 7
Results for Problem 2(c): $n = 20, n_z = 80, \gamma = 10^{-15}, N = 4, nzb = 10$.

M_{agg}	M_{disagg}	$Iter$	T_{total}	Err_{res}	Err_{rel}
GE_1	GE_1	1	0.02	$0.551E - 16$	$0.204E - 13$
GE_2	GE_2	1	0.02	$0.551E - 16$	$0.204E - 13$
GTH_2	GTH_2	1	0.02	$0.846E - 28$	$0.354E - 24$
GTH_1	GTH_1	1	0.02	$0.841E - 28$	$0.354E - 24$

(d) The parameters in this example are the same as those in 2(a) except

$$\eta_t + \eta = 10.$$

TABLE 8
Sizes of the aggregates for Problem 2(d): $\gamma = 10^{-3}$.

1	3	6	10	15	21	28
36	45	55	66			

TABLE 9
Results for Problem 2(d): $n = 286, n_z = 1{,}606, \gamma = 10^{-3}, N = 11, nzb = 31$.

M_{agg}	M_{disagg}	$Iter$	T_{total}	Err_{res}	Err_{rel}
GE_1	GE_1	8	0.47	$0.519E - 16$	$0.142E - 11$
GE_2	GE_2	8	0.64	$0.969E - 16$	$0.469E - 11$
GTH_2	GTH_2	8	1.18	$0.412E - 16$	$0.233E - 12$
GTH_1	GTH_1	8	2.72	$0.449E - 16$	$0.233E - 12$

(e) The parameters in this example are the same as those in 2(d).

TABLE 10
Sizes of the aggregates for Problem 2(e): $\gamma = 10^{-4}$.

65	55	165

TABLE 11
Results for Problem 2(e): $n = 286, n_z = 1{,}606, \gamma = 10^{-4}, N = 3, nzb = 7$.

M_{agg}	M_{disagg}	$Iter$	T_{total}	Err_{res}	Err_{rel}
GE_1	GE_1	5	0.88	$0.583E - 16$	$0.178E - 11$
GE_2	GE_2	5	1.09	$0.531E - 16$	$0.132E - 11$
GTH_2	GTH_2	5	2.89	$0.398E - 16$	$0.383E - 12$
GTH_1	GTH_1	5	13.84	$0.350E - 16$	$0.383E - 12$

(f) The parameters in this example are the same as those in 2(d) except

$$\mu_2 = (10^{-5})/30.$$

TABLE 12
Sizes of the aggregates for Problem 2(f): $\gamma = 10^{-3}$.

1	2	1	3	2	4	1	3	5	2	4	6	1	3	5	7
2	4	6	8	1	3	5	7	9	2	4	6	8	10		
1	3	5	7	9	11	2	4	6	8	10	1	3	5	7	9
2	4	6	8	1	3	5	7	2	4	6	1	3	5	2	4
1	3	2	1												

TABLE 13
Results for Problem 2(f): $n = 286, n_z = 1{,}606, \gamma = 10^{-3}, N = 66, nzb = 196$.

M_{agg}	M_{disagg}	$Iter$	T_{total}	Err_{res}	Err_{rel}
GE_1	GE_1	3	0.16	$0.247E-16$	$0.469E-13$
GE_2	GE_2	3	0.24	$0.247E-16$	$0.469E-13$
GTH_2	GTH_2	3	0.39	$0.111E-15$	$0.107E-14$
GTH_1	GTH_1	3	0.49	$0.111E-15$	$0.107E-14$

(g) The parameters in this example are the same as those in 2(a) except

$$\eta_t + \eta = 20 \text{ and } \lambda = (10^{-7})\eta_t.$$

TABLE 14
Sizes of the aggregates for Problem 2(g): $\gamma = 10^{-6}$.

1	3	6	10	15	21	28
36	45	55	66	78	91	105
120	136	153	171	190	210	231

TABLE 15
Results for Problem 2(g): $n = 1{,}771, n_z = 11{,}011, \gamma = 10^{-6}, N = 21, nzb = 61$.

M_{agg}	M_{disagg}	$Iter$	T_{total}	Err_{res}	Err_{rel}
GE_1	GE_1	3	3.79	$0.234E-16$	$0.263E-12$
GE_2	GE_2	3	4.11	$0.234E-16$	$0.263E-12$
GTH_2	GTH_2	3	15.12	$0.514E-18$	$0.605E-14$
GTH_1	GTH_1	3	82.47	$0.514E-18$	$0.605E-14$

(h) The parameters in this example are the same as those in 2(a) except

$$\eta_t + \eta = 20, \ \mu_1 = 0.2(10^{-10}), \text{ and } \mu_2 = (10^{-10})/30.$$

Sizes of the aggregates for Problem 2(h) for $\gamma = 10^{-12}$ are the same as for Problem 2(g).

TABLE 16
Results for Problem 2(h): $n = 1{,}771, n_z = 11{,}011, \gamma = 10^{-12}, N = 21, nzb = 61$.

M_{agg}	M_{disagg}	$Iter$	T_{total}	Err_{res}	Err_{rel}
GE_1	GE_1	3	2.23	$0.390E-16$	$0.466E-03$
GE_2	GE_2	3	2.88	$0.390E-16$	$0.466E-03$
GTH_2	GTH_2	3	9.12	$0.399E-16$	$0.583E-15$
GTH_1	GTH_1	3	26.89	$0.399E-16$	$0.584E-15$

In all problems considered, both IAD with GE and IAD with GTH converged (see $Iter$ in Table 1) with a residual of order machine epsilon regardless of the implementation scheme chosen. For each problem, the number of iterations taken by both methods is the same.

However, the relative error in IAD with GE is much larger than that of IAD with GTH in the first small problem (see Table 3) where IAD with GTH has a relative error almost of order machine epsilon. It is presumed this is due to the smaller size of the systems solved in this problem. The first test problem has a small coefficient matrix giving rise to a coupling matrix of size 3 × 3. Similarly, the largest system solved in the disaggregation step of the IAD algorithm is of size 3 × 3. Consequently, one expects to lose a relative accuracy of roughly 2 digits in this problem when IAD is used with GTH. The results confirm that IAD with GTH performs even better than the expectations, and only a single digit of accuracy is lost. However, as in the second problem, there are cases where the GTH algorithm does not achieve its worst-case bound due to statistical effects in rounding error accumulation although the matrices solved are considerably larger (see Tables 4, 8, 10, 12, and 14). Moreover, just as indicated in [10], it is possible to improve the entrywise relative error in the GTH algorithm even further by forming the pivot element in higher precision (quadruple precision for the given system parameters).

As expected, the number of iterations taken to achieve the prespecified tolerance criterion decreases as the decomposability parameter becomes smaller for a given problem. Among the problems considered, when the decomposability parameter is greater than or equal to 10^{-6}, IAD with GE provided results in which 3 to 5 digits of accuracy are lost (see Tables 3, 5, 9, 11, 13, and 15). An observation regarding the results of the second test problem is the difference between T_{total}'s for implementation Schemes 1 and 2 (see Tables 9, 11, 13, 15, and 16). First, although it is not the case for the second problem, the nonzero structure of a nonsymmetric matrix changes if the matrix is transposed, thus totally affecting the reduction process. Second, both methods in Scheme 2 store the multipliers, and thus they spend extra time. Finally, remember that Scheme 1 implementation of GTH is accomplished by shifting data around in memory, which is a very time-consuming process.

The results of the modified scheme (suggested by G. W. Stewart) are quite competitive with those of IAD with GTH for the first small problem (see Table 17). However, there are examples (Test Problem 2, parts (b), (c), (e), (f), (g)) with varying degrees of decomposability for which the modified scheme does not provide a relative error that is competitive with the relative error in IAD with GTH (see Tables 6, 7, 11, 13, and 15). However, it is never larger than the relative error in IAD with GE (see Table 17). The timing of the modified scheme is almost as good as that of IAD with GE even for the larger test cases.

6. Conclusion. In this paper, the computation of the stationary probability vectors of ill-conditioned NCD Markov chains is considered. The GTH method, which avoids subtractions, is a much more stable version of GE. For that reason, it is a good candidate to be used in the two-level IAD algorithm. Experiments on several problems are carried out, applying this

TABLE 17
Results for modified scheme (transposed version).

Problem	*Iter*	T_{total}	Err_{res}	Err_{rel}
1	4	0.04	$0.621E-16$	$0.423E-15$
2(a)	5	0.10	$0.458E-15$	$0.381E-12$
2(b)	1	0.07	$0.974E-17$	$0.345E-14$
2(c)	1	0.02	$0.551E-16$	$0.204E-13$
2(d)	8	0.56	$0.443E-16$	$0.245E-12$
2(e)	5	1.03	$0.385E-16$	$0.109E-11$
2(f)	3	0.28	$0.247E-16$	$0.469E-13$
2(g)	3	4.35	$0.234E-16$	$0.263E-12$
2(h)	3	2.71	$0.399E-16$	$0.273E-15$

idea versus GE in the IAD technique. The GTH approach to calculating the pivot element by taking the negated sum of the off-diagonal elements in the unreduced part of the pivot column proves to be valuable for singular M-matrices with 0 column sums, and it is shown to be quite effective on the problems of interest.

The GTH idea is employed to solve the linear systems of equations formed in both the aggregation and disaggregation steps of the IAD algorithm. For the first small problem considered, we observed that the relative error in IAD with GE is much larger than that of IAD with GTH, whose relative error is fixed in the order of machine epsilon. In all cases, the size of the systems solved provides an estimation for the worst case bound on the relative error for IAD with GTH. Additionally, just as explained in §4, it was not surprising to see IAD with GTH take on the order of 10 times as much time as IAD with GE for the larger chains in the second test problem. A modified scheme suggested by G. W. Stewart, which essentially performs diagonal correction on pivots only when there is a suspected loss in significance, seems to work very well in the IAD algorithm for small NCD Markov chains. The scheme possesses the good relative accuracy property of GTH and the convenient implementation of GE. However, for larger matrices, it is frequently no better than GE (while never being worse than GE) in terms of relative accuracy. It is also verified for IAD with GE that a significant difference between the relative error and the residual error is a clear indication of an ill-conditioned problem.

In conclusion, ordinary GE should definitely be avoided in both steps of the iterative IAD algorithm when solving NCD chains with a degree of coupling less than machine epsilon. However, if an approximation to the stationary vector of a large NCD Markov chain is sought in a short time, IAD with GE may be used. On the contrary, if relative error in the stationary vector of the NCD chain is deemed as of utmost importance, then IAD with GTH has to be recommended. However, we do recommend IAD with GE when the decomposability parameter for a given problem is greater than or equal to 10^{-6}. Only 3 to 5 digits of accuracy were lost in such problems considered. A compromise between GE and GTH seems to be the modified scheme suggested by G. W. Stewart. Examination of several sparse storage formats for both GTH and GE has indicated one disadvantage of the GTH method. The time to execute the IAD algorithm on large irreducible NCD Markov chains tends to be longer when the GTH method is used in the aggregation and disaggregation steps of the iterative solver. Memory requirement of the GTH algorithm is generally slightly higher than that of GE; nevertheless this mostly depends on the sparse storage format chosen. One last remark would be to direct attention to the possibility of exploiting the inherent parallelism in the formation of the coupling matrix. Similarly, parallel implementation of the solution of N nonsingular systems in the disaggregation step needs to be investigated.

REFERENCES

[1] A. Berman and R. J. Plemmons, *Nonnegative Matrices in the Mathematical Sciences*, Academic Press, New York, 1979.

[2] W. L. Cao and W. J. Stewart, *Iterative aggregation/disaggregation techniques for nearly uncoupled Markov chains*, J. Assoc. Comput. Mach., 32 (1985), pp. 702–719.

[3] P.-J. Courtois, *Decomposability: Queueing and Computer System Applications*, Academic Press, New York, 1977.

[4] W. K. Grassmann, M. I. Taksar, and D. P. Heyman, *Regenerative analysis and steady state distributions for Markov chains*, Oper. Res., 33 (1985), pp. 1107–1116.

[5] W. J. Harrod and R. J. Plemmons, *Comparison of some direct methods for computing the stationary distributions of Markov chains*, SIAM J. Sci. Statist. Comput., 5 (1984), pp. 453–469.

[6] D. P. Heyman, *Further comparisons of direct methods for computing stationary distributions of Markov chains*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 226–232.

[7] J. R. Koury, D. F. McAllister and W. J. Stewart, *Iterative methods for computing stationary distributions of nearly completely decomposable Markov chains*, SIAM J. Alg. Discrete Meth., 5 (1984), pp. 164–186.

[8] C. D. Meyer, *Stochastic complementation, uncoupling Markov chains, and the theory of nearly reducible systems*, SIAM Rev., 31 (1989), pp. 240–272.

[9] ———, *Sensitivity of the stationary distribution of a Markov chain*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 715–728.

[10] C. A. O'Cinneide, *Entrywise perturbation theory and error analysis for Markov chains*, Numer. Math., 65 (1993), pp. 109–120.

[11] C. C. Paige, P. H. Styan, and P. G. Wachter, *Computation of the stationary distribution of a Markov chain*, J. Statist. Comput. Simulation, 4 (1975), pp. 173–186.

[12] B. Philippe, Y. Saad, and W. J. Stewart, *Numerical methods in Markov chain modelling*, Oper. Res., 40 (1992), pp. 1156–1179.

[13] S. Pissanetzky, *Sparse Matrix Technology*, Academic Press, London, 1984.

[14] P. J. Schweitzer, *A survey of aggregation-disaggregation in large Markov chains*, in Numerical Solution of Markov Chains, W. J. Stewart, ed., Marcel Dekker, Inc., New York, 1991, pp. 63–88.

[15] H. Simon and A. Ando, *Aggregation of variables in dynamic systems*, Econometrica, 29 (1961), pp. 111–138.

[16] G. W. Stewart and G. Zhang, *On a direct method for the solution of nearly uncoupled Markov chains*, Numer. Math., 59 (1991), pp. 1–11.

[17] G. W. Stewart, W. J. Stewart, and D. F. McAllister, *A two-stage iteration for solving nearly completely decomposable Markov chains*, in The IMA Volumes in Mathematics and its Applications 60: Recent Advances in Iterative Methods, G. H. Golub, A. Greenbaum, and M. Luskin, eds., Springer-Verlag, New York 1994, pp. 201–216.

[18] G. W. Stewart, personal communication, 1994.

[19] W. J. Stewart, *A comparison of numerical techniques in Markov modelling*, Comm. ACM, 21 (1978), pp. 144–152.

[20] ———, *MARCA: Markov Chain Analyzer, A software package for Markov modeling*, in Numerical Solution of Markov Chains, W. J. Stewart, ed., Marcel Dekker, Inc., New York, 1991, pp. 37–61.

[21] W. J. Stewart and W. Wu, *Numerical experiments with iteration and aggregation for Markov chains*, ORSA J. Comput., 4 (1992), pp. 336–350.

[22] W. J. Stewart, *Introduction to the Numerical Solution of Markov Chains*, Princeton University Press, Princeton, NJ, 1994.

[23] Y. Takahashi, *A Lumping method for the Numerical Calculation of Stationary Distributions of Markov Chains*, Research report B-18, Dept. of Information Sciences, Tokyo Institute of Technology, 1975.

SIAM J. SCI. COMPUT.
Vol. 17, No. 2, pp. 305–327, March 1996

001

MOVING MESH METHODS FOR PROBLEMS WITH BLOW-UP*

CHRIS J. BUDD†, WEIZHANG HUANG‡, AND ROBERT D. RUSSELL§

Abstract. In this paper we consider the numerical solution of PDEs with blow-up for which scaling invariance plays a natural role in describing the underlying solution structures. It is a challenging numerical problem to capture the qualitative behaviour in the blow-up region, and the use of nonuniform meshes is essential. We consider moving mesh methods for which the mesh is determined using so-called *moving mesh partial differential equations* (MMPDEs). Specifically, the underlying PDE and the MMPDE are solved for the blow-up solution and the computational mesh simultaneously. Motivated by the desire for the MMPDE to preserve the scaling invariance of the underlying problem, we study the effect of different choices of MMPDEs and monitor functions. It is shown that for suitable ones the MMPDE solution evolves towards a (moving) mesh which close to the blow-up point automatically places the mesh points in such a manner that the ignition kernel, which is well known to be a natural coordinate in describing the behaviour of blow-up, approaches a constant as $t \to T$ (the blow-up time). Several numerical examples are given to verify the theory for these MMPDE methods and to illustrate their efficacy.

Key words. blow-up solution, moving mesh, scaling invariance

AMS subject classifications. 35K55, 65M50, 35B40, 65M20

1. Introduction.

Many mathematical idealizations of physical models have the property that they develop singularities in a finite time T. Examples are the blow-up of the solutions of models describing combustion in chemicals or chemotaxis in cellular aggregates and the formation of shocks in the inviscid Burgers' equation and the space-charge equations. Such a singularity often represents an important change in the properties of the model, such as the ignition of a heated gas mixture, and it is important that it should be accurately reproduced by a numerical computation.

When a singularity forms, changes occur on increasingly small length scales and, as the time T is approached, on increasingly smaller timescales. If a numerical method with a *fixed* mesh is used to reproduce such behaviour then its accuracy will diminish significantly when the length scale of the singularity approaches the spacing between mesh points. In some cases this will lead to numerical solutions which differ qualitatively from the underlying analytic solution. Indeed, examples can be found where a computation on a fixed mesh misses the blow-up entirely, or where the numerical solution blows up over the whole region even though the analytic solution develops a singularity at a single point [AB94].

To compute such singular behaviour accurately, it is essential to use a numerical method which adapts the spatial mesh as the singularity develops. Ideally, the numerical method will reproduce the singularity sufficiently accurately as $t \to T$ to mimic the asymptotic behaviour of the solution. A feature of a wide class of PDEs (partial differential equations) which makes this feasible is that the spatial structure of the singularity evolves in a fairly simple manner, often independent of any local structure in the initial conditions. Provided that the adaptive method can reproduce this simple asymptotic behaviour, there is reason to hope that a numerical scheme can be designed to perform accurately for all times $t < T$.

One class of problems which have this feature is the semilinear parabolic equations describing the blow-up of the temperature of a reacting medium, such as a burning gas. The

*Received by the editors February 14, 1994; accepted for publication November 28, 1994. This work was supported in part by NSERC grant OGP-0008781, the Nuffield Foundation, and SERC grant GR/J56219.

†School of Mathematics, University of Bristol, Bristol BS8 1TW, U.K. (Chris.Budd@bristol.ac.uk).

‡ Department of Mathematics and Statistics, Simon Fraser University, Burnaby, B.C. V5A 1S6, Canada. Current address: Department of Mathematics, 405 Snow Hall, University of Kansas, Lawrence, KS 66045 (whuang@math.ukans.edu).

§Department of Mathematics and Statistics, Simon Fraser University, Burnaby, B.C. V5A 1S6, Canada (rdr@cs.sfu.ca).

simplest equations describing such blow-up have the form

$$(1.1)\qquad u_t = u_{xx} + f(u), \quad u(0,t) = u(1,t) = 0, \quad u(0,x) = u_0(x),$$

where $f(u)$ is any convex function of u such that $f(u) \to \infty$ as $u \to \infty$. In this paper we shall consider the cases of $f(u) = u^p$, $f(u) = e^u$, and slightly more general problems. It is well known [FM85] that if $u_0(x)$ is "sufficiently large" and has a single nondegenerate maximum, then there is a blow-up time $T < \infty$ and a unique blow-up point x^* such that

$$(1.2)\qquad u(x^*, t) \to \infty \quad \text{as } t \to T$$

and

$$(1.3)\qquad u(x,t) \to u(x,T) < \infty \quad \text{if } x \neq x^*.$$

(If $t > T$ the solution becomes infinite everywhere.) Close to x^*, the solution $u(x,t)$ develops an isolated peak which becomes narrower, tending to zero width, as $t \to T$. A derivation and general study of these systems is given in [BE89].

The computation of the solutions of (1.1) is important for several reasons. First, although very simple, the formation of the singularities in this problem is typical of that of a wide class of PDEs modelling many differing physical phenomena. Second, a great deal is known about the analytic structure of the solutions of (1.1) for t close to T and x close to x^*, and thus they make excellent problems for testing the performance of and verifying the analysis for the numerical methods used in their solution. Third, having numerical methods which are faithful to the underlying asymptotics of the PDEs raises the possibility of solving very difficult problems for which the analytic structure is unknown, and then using the resulting numerical solutions to lend insight into this structure. Of course this can in turn motivate derivation of further theoretical results.

Existing adaptive numerical methods for solving (1.1) are described in [Cho81], [LPSS86], [BK88], [Ber89], and [BDS93]. These are either based upon closely exploiting the known analytic structure of the singularity or on an adaptive procedure which requires an increasingly larger number of mesh points to model the developing singularity as $t \to T$. In contrast, we shall describe here an elegant set of methods for solving (1.1) which use the dynamic gridding algorithms described in [HRR94b]. These MMPDE (moving mesh PDE) methods are based upon equidistributing a monitor function, say $M(u, u_x, u_{xx})$, which relies on no a priori knowledge of the solution to the PDE u, although an analysis of scaling properties of the PDE does lead to certain optimal choices of M. We shall show that these methods have the significant property that they reproduce the dynamical nature of the development of the singularity. In particular, there is a natural rescaling of the spatial coordinate close to the singularity which is automatically captured by the moving mesh method.

The asymptotic scaling of the singularity was first observed formally by [Dol85]. A full proof is given in [BB92]. They showed that for x close to x^* and t close to T there was a natural spatial coordinate, the so-called *ignition kernel* μ, where

$$(1.4)\qquad \mu(x,t) = (x - x^*)\left[(T-t)|\log(T-t)|\right]^{-\frac{1}{2}}.$$

In [BK88] the original PDE is recast in terms of the closely related "similarity" variable

$$(1.5)\qquad \zeta = \mu \ \log^{\frac{1}{2}}(T-t),$$

which is derived from a scaling invariance of the original PDE, and the resulting scaled PDE is then solved by using a static regridding algorithm. In contrast, we show that the MMPDE solution evolves towards a (moving) mesh which close to the singularity automatically places the mesh points $x_i(t)$ in such a manner that $\mu(x_i, t)$ approaches a constant as $t \to T$. In this way, the MMPDE methods naturally inherit the correct spatial structure of the singularity.

The paper is organized as follows. In §2 we introduce the blow-up problems and describe the asymptotic form of solutions close to blow-up. In §3 we introduce the MMPDE method for determining the mesh coordinate transformation used to solve these problems. Motivated by the desire to preserve scaling invariance, we consider the effect of different choices of the MMPDE and monitor function M. In §4 we show that in fact the "approximate" similarity solution behaviour of (1.1) is preserved by suitable choices of M, using techniques for the discrete analysis which mimic the previous analyses for the continuous case. In §5 we give numerical examples to verify the theory for these MMPDE methods and illustrate their efficacy. Finally, in §6 we state some conclusions and briefly discuss a more general framework under current investigation.

2. Structure of blow-up solutions for PDEs. We now consider the form of the solutions of the semilinear parabolic PDE

$$\begin{aligned} &u_t = u_{xx} + u^p, \quad p > 1, \\ &u(0,t) = u(1,t) = 0, \\ &u(x,0) = u_0(x) > 0. \end{aligned} \tag{2.1}$$

As well as looking at the asymptotic form of the solutions close to blow-up we also consider some of the underlying principles which lead to these solutions. These principles can then be used as a guide to the numerical method and also to study other related equations.

It has been shown by several authors, e.g., [FM85], that if $u_0(x)$ is sufficiently large and positive and has a single nondegenerate maximum, then (1.2) and (1.3) hold. The point x^* and the time T depend subtlely upon $u_0(x)$ but, remarkably, the solution $u(x,t)$ itself is almost independent of u_0 provided that x and t are close to x^* and T, respectively. The blow-up profile takes the form of an isolated spike of increasingly narrow width and has been studied by [Dol85] and [BK88]. The behaviour of this spike may be described as follows.

THEOREM 2.1. *Let $\beta = \frac{1}{p-1}$ and let $\mu(x,t)$ be defined by*

$$\mu = (x - x^*)\left[(T-t)(\alpha - \log(T-t))\right]^{-\frac{1}{2}} \tag{2.2}$$

where α is a constant which depends on $u_0(x)$.

(i) *If $x(t)$ is taken to keep $\mu(x,t)$ constant, then the solution $u(x,t)$ to* (2.1) *satisfies*

$$(T-t)^\beta u(x,t) \to \beta^\beta \left[1 + \frac{\mu^2}{4p\beta}\right]^{-\beta} \quad \text{as } t \to T. \tag{2.3}$$

(ii) *If $|x - x^*|$ is small but fixed and independent of t, then*

$$u(x,t) \to u(x,T) = \left[4p\beta^2 \frac{|\alpha - 2\log|x-x^*||}{|x-x^*|^2}\left(1 + O(|x-x^*|)^2\right)\right]^\beta \tag{2.4}$$

as $t \to T$.

We observe that the expressions (2.3) and (2.4) coincide if we set μ to be large. The expression (2.3) describes the evolution of the blow-up peak in terms of the "ignition kernel" μ. This variable was first identified by [Dol85], and is a natural variable to describe the spatial structure of blow-up. A remarkable feature of the numerical methods we shall describe is that close to x^* the moving mesh is placed precisely at those points for which μ is constant.

The starting point for deriving these expressions is a natural scaling invariance of the solutions of (2.1) (in the absence of boundary conditions). In particular, the equation is invariant under the scaling

$$\left\{\begin{array}{l}(T-t)\to\lambda(T-t),\\ u\to\frac{1}{\lambda^\beta}u,\\ x\to\lambda^{\frac{1}{2}}x\end{array}\right. \tag{2.5}$$

for any positive λ. A similarity solution of (2.1) is any solution which is invariant under this scaling. Many interesting PDEs, including problems leading to blow-up (see below), have a scaling invariance similar to (2.5).

Motivated by (2.5) we recast (2.1) in terms of *similarity variables* $w(s,y)$, y, and s defined by

$$\left\{\begin{array}{l}s=-\log(T-t),\\ w(s,y)=(T-t)^\beta u(x,t),\\ y=(x-x^*)(T-t)^{-1/2}\end{array}\right. \tag{2.6}$$

to give the partial differential equation

$$w_s=w_{yy}-\frac{1}{2}yw_y+w^p-\beta w \tag{2.7}$$

supplemented with

$$w_y(s,0)=0 \text{ and } w(s,y)\to 0 \text{ as } |y|\to\infty. \tag{2.8}$$

The latter condition is necessary to match the boundary conditions satisfied by the solutions of the unscaled problem.

A similarity solution of the original PDE is a steady state (i.e., s independent) solution of (2.7) which also satisfies (2.8). These solutions were originally proposed as solutions of (2.1), but in fact, the only bounded, nonzero steady state solution of (2.7) is

$$w(s,y)=\beta^\beta, \tag{2.9}$$

which fails to satisfy (2.8). However, if we consider a set of points $x(t)$ such that $y(x,t)$ is fixed, then by using energy arguments Giga and Kohn [GK85] show that

$$(T-t)^\beta u(x,t)\to\beta^\beta, \tag{2.10}$$

so that the function (2.9) is an attractor for solutions of (2.7) over compact sets in y.

To calculate a solution of (2.7) which corresponds to a solution of (2.1) we consider instead a perturbation of the similarity solution by setting

$$w(s,y)=f\left(\frac{y}{g(s)}\right)\equiv f(z) \tag{2.11}$$

where f and g satisfy the conditions

$$\left\{\begin{array}{l}f(z)\to 0 \text{ as } |z|\to\infty,\\ g(s)\to\infty \text{ as } s\to\infty,\\ \frac{g'}{g}\to 0 \text{ as } s\to\infty.\end{array}\right. \tag{2.12}$$

The derivation of these conditions is given in [BK88]. Substituting into (2.7) gives

$$-\frac{g'}{g}zf_z - g^{-2}f_{zz} + \frac{1}{2}zf_z + \beta f - f^p = 0 \tag{2.13}$$

which, for large s, reduces to the first-order equation

$$\frac{1}{2}zf_z + \beta f - f^p = 0 \tag{2.14}$$

with solution

$$f(z) = \left[\frac{1}{\beta} + cz^2\right]^{-\beta}, \tag{2.15}$$

where c is a constant which without loss of generality may be set to 1. The reduction of problem (2.1) from a second-order equation (2.7) to a first-order Hamilton-Jacobi one (2.14) is a crucial feature of blow-up problems, see [GV93]. The function g, giving the spreading rate, can be derived formally by making an expansion of the function w in powers of $\frac{1}{s}$ and matching terms. This implies that g^{-2}, $\frac{g'}{g}$, and s^{-1} should all be of the same order for $s >> 1$. The details of this derivation are given in [BK88] and [Dol85] and give

$$g(s) = \left[4\beta^2 p(s+\alpha)\right]^{\frac{1}{2}} \tag{2.16}$$

where α is a constant which depends (weakly) upon the initial conditions. Combining (2.15) with (2.16) and taking s large gives (2.3).

The derivation given here is formal and is closely related to an analysis of the behaviour of the numerical scheme described in §4. A more precise derivation of the result (2.3) describing the shape of the blow-up peak follows from a centre-manifold reduction of the solutions of (2.7) and is given in [BB92], [FK92], and [HV93]. These papers also give a rigorous derivation of (2.4), although this equation follows formally from taking the large μ limit of (2.3) and matching to a steady state solution.

A useful conclusion from (2.3) is the natural relationship between the various scalings involved in the solution of the blow-up problem. In particular, if we consider $\tau = (T - t)$ to be the local "timescale" for blow-up, then the corresponding scale for u is

$$U = \frac{1}{\tau^\beta}, \tag{2.17}$$

and the length scale for x is

$$X = \left[\tau(\alpha - \log\tau)\right]^{1/2}. \tag{2.18}$$

Furthermore, the scale for u_x is approximately given by

$$u_x \approx \frac{U}{X} = \left[\tau^{1+2\beta}(\alpha - \log\tau)\right]^{-1/2}. \tag{2.19}$$

These scales are useful in deciding the choice of an appropriate numerical method. For example, (2.18) gives an indication of the correct mesh spacing close to the blow-up point.

The scaling invariance (2.5) plays a crucial role in the analysis of (2.1). Although the solution of the PDE is not self-similar, it is close to being self-similar and moreover converges to the self-similar solution $w = \beta^\beta$ on any compact interval in the similarity variable y.

Such scaling invariance and associated self-similar and approximately self-similar solution behaviour is found in many other equations describing blow-up and for completeness we list some of these here. Further examples of semilinear and quasi-linear PDEs related to (2.1) include the Kassoy problem [Kas77], [Gel63]

$$u_t = u_{xx} + e^u, \tag{2.20}$$

and porous-medium reaction-diffusion equations

$$u_t = (|u_x|^\sigma u_x)_x + e^u \tag{2.21}$$

and

$$u_t = (u^\alpha u_x)_x + u^p, \tag{2.22}$$

which are analyzed in [BDG93] and in [GP91], respectively. Furthermore, the nonlinear Schrödinger equation [LPSS86]

$$i\psi_t + \Delta\psi + |\psi|^{2\sigma}\psi = 0 \tag{2.23}$$

is an example of a hyperbolic PDE for which blow-up occurs in R^N if $N \geq 2$, and this has been used to model focusing in lasers.

The similarity variables for (2.20) and (2.21) are

$$\begin{cases} w(s, y) = \log(T - t) + u(x, t), \\ y = x(T - t)^{-\frac{1}{\sigma+2}}, \end{cases} \tag{2.24}$$

and with this change of variables (2.20) and (2.21) reduce to PDEs similar to (2.7), which can be analyzed in an analogous way. Remarkably, the corresponding solution is approximately self-similar if $\sigma = 0$ and exactly self-similar if $\sigma > 0$. Similar behaviour to that in (2.1) is observed for (2.22) with $\alpha > 0$, where

$$y = x(T - t)^{-\frac{1}{2}[1+\alpha/(p-1)]}. \tag{2.25}$$

The behaviour of the nonlinear Schrödinger equation is less well understood, but it is invariant under various scalings. In particular, it has a natural set of similarity variables given by

$$\begin{cases} y = \frac{r}{L(t)}, \\ \tau = \int_0^t \frac{ds}{L}, \\ u(y, \tau) = L^{1/\sigma}\psi, \end{cases} \tag{2.26}$$

where $r = |x|$ and $L(t)$ can be chosen in various ways. These follow from the scaling invariance of solutions given by

$$\psi(x, t) \to \lambda^{-1/\sigma}\psi\left(\frac{x}{\lambda}, \frac{t}{\lambda^2}\right). \tag{2.27}$$

In [LPSS86] extensive use is made of this rescaling in calculating the solution of (2.23) numerically. Self-similar solutions arise when $L(t) = (T - t)^{1/2}$, as in the previous problems, and numerical evidence for the existence of such solutions has been obtained.

It is clear from this brief discussion that the scaling invariance of (2.1) and (2.20) - (2.23) plays a key role in determining the dynamical solution behaviour. This strongly implies that numerical methods which respect this invariance should be more effective in reproducing the dynamics than those which do not. Such methods must necessarily employ moving meshes to allow for rescalings in both space and the solution. We now consider these.

3. Moving mesh PDEs with scaling invariance. The class of methods which we propose for solving (1.1) are the moving mesh PDE methods described in [HRR94a] and [HRR94b]. For these the function $u(x,t)$ is discretized to give the solution values $u_i(t)$ defined on a moving mesh $x_i(t)$, $i = 0, \dots, N$. The boundary conditions in (1.1) give $u_0 = u_N = 0$ and $x_0 = 0$, $x_N = 1$. The mesh $x_i(t)$ is defined in terms of a differentiable mesh transformation $x(\xi, t)$: $[0,1] \to [0,1]$, where x is the physical coordinate and ξ is the computational coordinate such that

$$x_i(t) = x\left(\frac{i}{N}, t\right). \tag{3.1}$$

In the continuous formulation, the constraint

$$\frac{\partial x}{\partial \xi} > 0 \tag{3.2}$$

assures that the mesh transformation is well defined for fixed t, and the discrete analogue is that mesh crossing does not occur. For the MMPDE approach, a new partial differential equation for $x(\xi, t)$, called the moving mesh PDE, is solved simultaneously with the original PDE for $u(x,t)$. The underlying strategy for determining $x(\xi,t)$ is to require *equidistribution* of a positive *monitor function*, say $M(u, u_x, u_{xx})$, so that

$$\int_0^x M dy = \xi \int_0^1 M dy. \tag{3.3}$$

Equivalently, differentiating this identity gives

$$\frac{\partial}{\partial \xi}\left(M \frac{\partial x}{\partial \xi}\right) = 0, \; x(0,t) = 0, \; x(1,t) = 1. \tag{3.4}$$

A mesh (or a coordinate transformation) is said to be *equidistributed* when (3.4) holds. It is convenient in practice not to enforce exact equidistribution upon a mesh but to instead solve an MMPDE for which it tends toward an equidistributed state. This has the advantages that a simple initial mesh (such as a uniform one) can be used, the process produces stable meshes with less risk of mesh crossing than if (3.3) were enforced, and combined with a smoothing approach it reduces the problem (associated, for example, with the schemes proposed in [LPSS86]) of placing so many points close to the developing singularity that resolution is lost elsewhere. Of the various MMPDEs proposed in [HRR94a], we consider the two labelled MMPDE4 and MMPDE6. These are, respectively,

$$\tau \frac{\partial}{\partial \xi}\left(M \frac{\partial \dot{x}}{\partial \xi}\right) = -\frac{\partial}{\partial \xi}\left(M \frac{\partial x}{\partial \xi}\right) \tag{MMPDE4}$$

and

$$\tau \frac{\partial^2 \dot{x}}{\partial \xi^2} = -\frac{\partial}{\partial \xi}\left(M \frac{\partial x}{\partial \xi}\right). \tag{MMPDE6}$$

Here, $\dot{x}$ denotes $\frac{\partial x}{\partial t}|_{\xi \text{ fixed}}$ and τ is a small parameter which acts to relax the mesh to the equidistributed state. Note that MMPDE6 is the second derivative with respect to ξ of the integro-differential equation

$$\tau \dot{x} = -\left(\int_0^x M dx - \xi \int_0^1 M dx\right). \tag{3.5}$$

It is worth remarking that many previous moving mesh equations can be regarded as variants of or discrete approximations of these MMPDEs. (See, e.g., [HL89] for MMPDE6; see also [HRR94a] and [HRR94b].)

If we apply moving mesh PDE methods to solve a PDE with an underlying scaling invariance, then as the underlying PDE and the MMPDE are solved simultaneously to determine both the solution u and the mesh x, it is desirable that the scaling invariance of the underlying PDE be preserved. For (2.1) this requires that MMPDE4 or MMPDE6 be invariant under the scaling (2.5). The parameter τ and the monitor function M can indeed be suitably chosen to meet this requirement. Assuming that the solution $u(x, t)$ to (2.1) is positive for $x \in (0, 1)$ and $t \geq 0$, MMPDE6 can be made invariant under the scaling (2.5) if τ is taken as a dimensionless constant and

$$M(u) = u^{p-1}. \tag{3.6}$$

However, regardless of the choice of M, τ cannot be constant if MMPDE4 is to be invariant under (2.5) (although we can obtain an invariance if τ is chosen adaptively).

This difference between MMPDE6 and MMPDE4 becomes important when we consider the timescales under which the mesh adapts to follow the structure of the solution. An inspection of MMPDE4 and MMPDE6 shows that each has a natural timescale T_{mesh} for adapting the mesh towards an equidistributed mesh. For MMPDE4 $T_{\text{mesh}} = O(\tau)$ and for MMPDE6 $T_{\text{mesh}} = O(\frac{\tau}{M})$. If T_{mesh} is significantly greater than the natural timescale for the evolution of the solution structure, then the mesh cannot adapt rapidly enough to follow the solution structure and is to all intents and purposes fixed. As we showed in §2, the natural timescale for the evolution of the blow-up peak is $O(T - t)$. Hence, if

$$T - t << \tau, \tag{3.7}$$

then MMPDE4 will not be able to evolve the mesh rapidly enough to follow the evolution of the peak. In contrast, if we look at MMPDE6 with M as in (3.6), then $\frac{\tau}{M} = \frac{\tau}{u^{p-1}} = \tau(T - t)$. Hence the timescale for the evolution of the mesh is always a factor τ smaller than the natural timescale of the underlying problem, and MMPDE6 will continue to evolve the mesh for t close to T. This is a direct consequence of the scaling invariance of MMPDE6 under this choice of monitor function. Other choices of monitor function (for example, arclength) do not share this property, and we will study this in more detail in §4.

For the numerical computation, PDE (2.1) is transformed in terms of the computational coordinate ξ and discretized by central finite differences on a uniform mesh in the computational domain. That is, (2.1) is first transformed into the quasi-Lagrangian form

$$\begin{cases} \dot{u} - \frac{u_\xi}{x_\xi}\dot{x} = \frac{1}{x_\xi}\left(\frac{u_\xi}{x_\xi}\right)_\xi + u^p, \\ u(0, t) = u(1, t) = 0, \end{cases} \tag{3.8}$$

and discretization gives the equation

$$\dot{u}_i - \frac{u_{i+1} - u_{i-1}}{x_{i+1} - x_{i-1}}\dot{x}_i = \frac{2}{x_{i+1} - x_{i-1}}\left(\frac{u_{i+1} - u_i}{x_{i+1} - x_i} - \frac{u_i - u_{i-1}}{x_i - x_{i-1}}\right) + u_i^p, \tag{3.9}$$

$i = 1, \ldots, N - 1$ and $u_0 = u_N = 0$. Similar finite difference equations can be obtained for discretizations of MMPDEs 4 and 6.

It is well known that for moving mesh methods, some sort of smoothing of the mesh is often necessary in order to obtain nonoscillatory, reasonably accurate solutions (e.g., see [DD87], [FVZ90], and [HRR94b]). In [DD87], Dorfi and Drury use a technique which smooths the node concentration defined by $\frac{1}{x_{i+1}-x_i}$. In [VBFZ89], Verwer et al. prove that smoothing the node concentration is basically equivalent to smoothing the monitor function over all points. To maintain the local structure of the underlying difference equations, we use the technique employed in [HRR94b]. Specifically, the values of the smoothed monitor function $\tilde{M}$ at nodes are defined by

$$\tilde{M}_i = \frac{\sum_{k=i-i_p}^{i+i_p} M_k \left(\frac{\gamma}{\gamma+1}\right)^{|k-i|}}{\sum_{k=i-i_p}^{i+i_p} \left(\frac{\gamma}{\gamma+1}\right)^{|k-i|}}, \tag{3.10}$$

where $M_i \equiv M(\xi_i, t)$, i_p is a nonnegative integer, and γ is a positive constant. The summations in (3.10) are understood to contain only elements with indices in the range between 0 and N. Notice that the replacement of M_i by $\tilde{M}_i$ is basically equivalent to using a smoother monitor function, and $i_p = 0$ corresponds to the nonsmoothing case. Values of the parameters γ and i_p need to be selected for these moving mesh PDE methods. In our experience (also see [HRR94b]), the choice of γ is fairly insensitive and can generally be fixed. In this paper, we use $\gamma = 2$. The value for i_p usually is taken as 0, 1, 2, 3, or 4.

The final forms for the discrete moving mesh equations for MMPDE4 and MMPDE6 are

$$\begin{aligned} &\tau\left(\tilde{M}_{i+\frac{1}{2}}(\dot{x}_{i+1} - \dot{x}_i) - \tilde{M}_{i-\frac{1}{2}}(\dot{x}_i - \dot{x}_{i-1})\right) \\ &= -\left(\tilde{M}_{i+\frac{1}{2}}(x_{i+1} - x_i) - \tilde{M}_{i-\frac{1}{2}}(x_i - x_{i-1})\right), \end{aligned} \tag{3.11}$$

and

$$\tau\left(\dot{x}_{i+1} - 2\dot{x}_i + \dot{x}_{i-1}\right) = -\left(\tilde{M}_{i+\frac{1}{2}}(x_{i+1} - x_i) - \tilde{M}_{i-\frac{1}{2}}(x_i - x_{i-1})\right), \tag{3.12}$$

$i = 1, \dots, N-1$ supplemented with $x_0 = 0$, $x_N = 1$, where $\tilde{M}_{i+\frac{1}{2}} := (\tilde{M}_i + \tilde{M}_{i+1})/2$. We note that the smoothing process maintains the dimension of the monitor function, which will become important when we consider rescalings of the equations.

4. Analysis of the solutions of the moving mesh equations.

We now analyze the solution behaviour for the discretization of problem (2.1), using the discrete version of MMPDE6 together with the monitor function given by (3.6). The analysis will be in two parts. First, we shall solve MMPDE6 exactly assuming that $u(x, t)$ is as described in §2. This will, in effect, determine the "optimum" mesh for such a problem. Second, we shall analyze the solutions of the coupled finite difference equations (3.9) and (3.12), assuming exact time integration, and compare the solutions with those obtained in the continuous case.

4.1. Analysis of the moving mesh equations.

Integrating MMPDE6 with respect to ξ, we know that the mesh transformation $x(\xi, t)$ satisfies the equation

$$-\tau\dot{x}_\xi = Mx_\xi + \theta(t) \tag{4.1}$$

where $\theta(t)$ is an integral constant which will be determined by the boundary conditions for $x(\xi, t)$. Indeed, integrating (4.1) with respect to ξ we have

$$-\tau\dot{x}(\xi, t)|_{\xi=0}^{1} = \int_0^1 Mx_\xi d\xi + \theta(t). \tag{4.2}$$

Then, from the boundary conditions

$$x(0,t) = x(1,t) - 1 = 0, \tag{4.3}$$

(4.2) implies that

$$\theta(t) = -\int_0^1 M dx. \tag{4.4}$$

Because the function $u(x,t)$ has a sharp peak, the integral above is asymptotically dominated by the contribution from the peak. It follows from (3.6) and (2.3) that within the blow-up region M has the asymptotic form

$$M = \frac{\beta}{(T-t)\left[1+\frac{\mu^2}{4p\beta}\right]}, \tag{4.5}$$

where $\mu(x,t)$ is defined in (2.2). If we take ϵ to be a small (fixed) positive value, we then have that

$$\begin{aligned} \theta(t) &\approx -\int_{x^*-\epsilon}^{x^*+\epsilon} \frac{\beta}{(T-t)\left[1+\frac{\mu^2}{4p\beta}\right]} dx \\ &\approx -\frac{\beta[\alpha-\log(T-t)]^{1/2}}{(T-t)^{1/2}} \int_{-\epsilon[\alpha-\log(T-t)]^{-1/2}(T-t)^{-1/2}}^{\epsilon[\alpha-\log(T-t)]^{-1/2}(T-t)^{-1/2}} \frac{dy}{1+\frac{y^2}{4p\beta}}. \end{aligned} \tag{4.6}$$

As $t \to T$ the limits in the above integral tend to $\pm\infty$ so that

$$\begin{aligned} \theta(t) &\approx -\frac{\beta[\alpha-\log(T-t)]^{1/2}}{(T-t)^{1/2}} \int_{-\infty}^{\infty} \frac{dy}{1+\frac{y^2}{4p\beta}} \\ &\approx -\frac{\pi\beta\sqrt{4p\beta}[\alpha-\log(T-t)]^{1/2}}{(T-t)^{1/2}}. \end{aligned} \tag{4.7}$$

The contribution to $\theta(t)$ from the range outside that considered will be of $O(1)$ in time and will be asymptotically dominated by the estimate in (4.7). It then follows that within the blow-up region (4.1) reads as

$$-\tau \dot{x}_\xi = \frac{\beta}{(T-t)(1+\frac{\mu^2}{4p\beta})} x_\xi - \frac{\pi\beta\sqrt{4p\beta}[\alpha-\log(T-t)]^{1/2}}{(T-t)^{1/2}}. \tag{4.8}$$

The definition of $\mu(x,t)$ in (2.2) suggests an ansatz for the mesh behaviour of the form

$$x(\xi,t) = x^* + (T-t)^{1/2}[\alpha - \log(T-t)]^{1/2} z(\xi). \tag{4.9}$$

Substituting this into (4.8) gives

$$-\tau z_\xi \left[-\frac{1}{2} + O\left(\frac{1}{\log(T-t)}\right)\right] = \frac{\beta z_\xi}{1+\frac{z^2}{4p\beta}} - \pi\beta\sqrt{4p\beta}. \tag{4.10}$$

The choice of the monitor function to be u^{p-1} ensures that the left-hand side and the first term of the right-hand side of (4.1) have the same scale in $(T-t)$. The term $O\left(\frac{1}{\log(T-t)}\right)$ on the left-hand side of (4.10) emphasizes the difference between an exactly self-similar and an approximately self-similar solution. Since $z(\xi)$ should be finite in the blow-up region, the left-hand side term is small compared with the first term on the right-hand side if $\tau << 1$ and therefore can be dropped. Hence, to the approximation of order τ we obtain an asymptotic equation within the blow-up region as

$$\frac{\beta z_\xi}{1+\frac{z^2}{4p\beta}} - \pi\beta\sqrt{4p\beta} = 0 \tag{4.11}$$

or

$$\beta\sqrt{4p\beta}\,\tan^{-1}\left(\frac{z}{2\sqrt{p\beta}}\right) = \pi\beta\sqrt{4p\beta}(\xi-\xi^*), \tag{4.12}$$

where we have assumed that $z = 0$ when $\xi = \xi^*$, i.e., blow-up occurs at the point $x^* = x(\xi^*, T)$.

Equation (4.12) describes the distribution of the mesh points within the peak, and the boundary conditions (4.3) determine ξ^*. From (4.9) these two conditions correspond in the limit of large $|z|$, i.e.,

$$\begin{cases} z \to -\infty, & \text{as } \xi \to 0, \\ z \to +\infty, & \text{as } \xi \to 1, \end{cases} \tag{4.13}$$

implying

$$\xi^* = \frac{1}{2}. \tag{4.14}$$

Our choice of M implies that the terms of order τ will be consistently small inside the blow-up peak.

Combining these results, we deduce that the mesh function x is given by

$$x(\xi,t) = x^* + 2\sqrt{\beta p}(T-t)^{1/2}[\alpha - \log(T-t)]^{1/2}\,\tan\left(\pi(\xi-\frac{1}{2})\right) + O(\tau) \tag{4.15}$$

so that

$$\frac{x(\xi,t)-x^*}{(T-t)^{1/2}[\alpha-\log(T-t)]^{1/2}\,\tan\pi(\xi-\frac{1}{2})} \to 2\sqrt{\beta p} \text{ as } t \to T. \tag{4.16}$$

Substituting this into the expression for $u(x,t)$ gives

$$u(\xi,t) = (T-t)^{-\beta}\beta^{\beta}[\cos(\pi(\xi-1/2))]^{2\beta}. \tag{4.17}$$

This expression for u in the computational domain may be easily checked against numerical calculations.

The above analysis only applies for mesh points within the peak. However, as $\xi \to 0$ or $\xi \to 1$ the mesh points must eventually tend to 0 and 1, respectively, and close to $\xi = 0$ they correspond to points where $u(x,t)$ is not changing rapidly and hence is bounded as $t \to T$. If we fix x, then as $t \to T$,

$$u(\xi,t) \to \left[\frac{8\beta^2 p\,|\log|x-x^*||}{(x-x^*)^2}\right]^{\beta} + O(x-x^*)^2. \tag{4.18}$$

Thus,

$$\int_0^x u^{1/\beta} \approx 8\beta^2 p \int_0^x \frac{|\log|x - x^*||}{(x - x^*)^2} \approx 8\beta^2 p \frac{|\log|x - x^*||}{(x - x^*)^2}. \tag{4.19}$$

Similarly, from (4.11),

$$\int_0^1 u^{1/\beta} = \theta(t) \approx \frac{\beta\pi\sqrt{4p\beta}\,[\alpha - \log(T - t)]^{1/2}}{(T - t)^{1/2}}, \tag{4.20}$$

so that an equidistributed mesh will give

$$\int_0^x u^{1/\beta} \approx \theta(t)\xi, \tag{4.21}$$

and hence

$$8\beta^2 p \frac{|\log|x - x^*||}{(x - x^*)^2} \approx \frac{\beta\pi\sqrt{4p\beta}\,[\alpha - \log(T - t)]^{1/2}\xi}{(T - t)^{1/2}}. \tag{4.22}$$

After some manipulation we find that this is precisely the form that (4.16) takes as $\xi \to 0$ (or indeed as $\xi \to 1$).

We observe from (4.15) and (4.22) that x is close to x^* if ξ lies in the interval $[\epsilon, 1 - \epsilon]$ where $\epsilon = (T - t)^{1/2}[\alpha - \log(T - t)]^{1/2}$. Outside this interval, x rapidly tends to 0 or 1. The implications of this analysis are that if $\Delta\xi$ is the mesh spacing in the computational domain, then a significant number of mesh points will be away from the peak only if

$$\Delta\xi < (T - t)^{1/2}[\alpha - \log(T - t)]^{1/2}. \tag{4.23}$$

A similar calculation can be made for the arclength monitor function

$$M = \sqrt{1 + u_x^2}. \tag{4.24}$$

Indeed, when u_x is large, $M \approx u_x$, and we have from (4.1), (4.4), and (4.9) that

$$\begin{aligned} &-\tau z_\xi \left[-\tfrac{1}{2} + O\left(\tfrac{1}{\log(T-t)}\right)\right] [\alpha - \log(T - t)]^{1/2} (T - t)^{\beta - 1/2} \\ &= \frac{2\beta^{\beta+1}|z|z_\xi}{4p\beta(1+\frac{z^2}{4p\beta})^{\beta+1}} + \Theta, \end{aligned} \tag{4.25}$$

where Θ is a constant. Thus, when $\tau[\alpha - \log(T - t)]^{1/2}(T - t)^{\beta-1/2} << 1$, for which we require $p < 3$ and τ to be fixed and small, then within the blow-up region the mesh equation is asymptotic to

$$\frac{2\beta^{\beta+1}|z|z_\xi}{4p\beta(1 + \frac{z^2}{4p\beta})^{\beta+1}} + \Theta = 0, \tag{4.26}$$

which leads to

$$u(\xi, t) = (T - t)^{-\beta}\beta^\beta \left[1 - 2|\xi - \xi^*|\right]. \tag{4.27}$$

However, when $\tau[\alpha - \log(T - t)]^{1/2}(T - t)^{\beta-1/2} >> 1$, which occurs when $p > 3$ and t is sufficiently close to T, then the left-hand side of (4.25) dominates over other terms and then there exists no solution satisfying the resulting asymptotic equation and the matching conditions (4.13). This implies that the mesh (or the coordinate transformation) cannot take the form (4.9) and hence is not "optimal" when $\tau[\alpha - \log(T-t)]^{1/2}(T-t)^{\beta-1/2} >> 1$. Indeed, in this case the timescale for the mesh evolution is greater than $T-t$ and the mesh ceases to evolve. To be precise, this occurs when

$$\tau(T-t)^{\beta-1/2} = \tau(T-t)^{(3-p)/2(p-1)} = \tau u_{\max}^{(p-3)/2} >> 1. \tag{4.28}$$

Finally we note that the above analysis is general and can also be applied to MMPDE4. Since the argument is quite similar, we omit it here but refer the reader to the discussion in §3.

4.2. The analysis of the discrete equations. We now turn our attention to an analysis of the discretizations (3.9) and (3.12) of the PDEs for blow-up and mesh evolution. We show that these equations admit a discrete solution which evolves in an "approximately self-similar" manner very closely related to that of the solution described in §2. Although we do not prove that this discrete solution is an attractor, the numerical calculations given in §5 strongly imply that it is, and the asymptotic predictions of this section agree with the observations in §5. (We emphasize at this point that the numerical solution may have a slightly different blow-up time from the analytical solution. However, as we are interested in the dynamics close to blow-up rather than the blow-up time itself we shall treat the blow-up times of the analytic and numerical methods as the same.)

Since the discrete equations have the same scaling invariance as the continuous ones, it is reasonable to consider a discrete solution in terms of similarity variables closely related to those in (2.6), namely

$$\begin{cases} w_i = (T-t)^{\beta} u_i, \\ y_i(s) = (x_i - x^*)(T-t)^{-1/2}, \\ s = -\log(T-t). \end{cases} \tag{4.29}$$

In these similarity variables we consider solutions with w_i independent of time. Substituting into (3.9) gives

$$\begin{aligned} &\beta w_i + \tfrac{w_{i+1}-w_{i-1}}{y_{i+1}-y_{i-1}}\left(\tfrac{1}{2}y_i - \dot{y}_i\right) \\ &= \tfrac{2}{y_{i+1}-y_{i-1}}\left(\tfrac{w_{i+1}-w_i}{y_{i+1}-y_i} - \tfrac{w_i-w_{i-1}}{y_i-y_{i-1}}\right) + w_i^p. \end{aligned} \tag{4.30}$$

The equation (4.30) is essentially a discretization of (2.7) on the nonuniform mesh $\{y_i\}$ and must also satisfy a discrete form of the boundary conditions (2.8). Like (2.7) it also possesses discrete "similarity solutions" for which both w_i and y_i are independent of time, and these solutions satisfy a discretization of the steady state of (2.7). However, this latter equation has only one solution which does not grow exponentially with y, and that is the constant solution, which does not satisfy the boundary conditions. (We note that $w_i \equiv \beta^{\beta}$ is also a solution of (4.30).) Provided that the mesh $\{y_i\}$ is sufficiently fine we will expect that the (nonconstant) discrete solutions of the steady state of (4.30) should approximate their continuous counterparts and also grow rapidly with i. Such solutions will not then match the boundary conditions. We can conclude from this reasoning that (4.30) is unlikely to have a steady state solution and that we should instead seek a solution which, like its analytical counterpart, is approximately self-similar rather than self-similar. We can compute these as in §2 by setting

$$y_i(s) = g(s)z_i \tag{4.31}$$

where z_i is independent of time and the function g satisfies (cf. (2.12))

$$g(s) \to \infty \text{ and } \frac{g'}{g} \to 0 \text{ as } s \to \infty. \tag{4.32}$$

Substituting into (4.30) and letting $s \to \infty$, we have

$$\beta w_i + \frac{w_{i+1} - w_{i-1}}{z_{i+1} - z_{i-1}} \frac{z_i}{2} = w_i^p. \tag{4.33}$$

This is a centred difference discretization of (2.14), and hence, provided the mesh $\{z_i\}$ is sufficiently regular,

$$w_i = \left(\frac{1}{\beta} + z_i^2\right)^{-\beta} + \epsilon_i, \tag{4.34}$$

where ϵ_i is a small error which decreases with the number of mesh points. As before, the form of g for large s can be derived from formal scaling arguments which give

$$g(s) = c[\alpha + s]^{1/2}. \tag{4.35}$$

We can repeat the arguments in [BK88] to deduce the value of the constant c which, to within a small error similar to ϵ_i, is consistent with (2.16). This demonstrates that an approximately self-similar solution for the numerical scheme (4.30) exists which has the correct dependence upon z_i and hence upon x_i. Finally, we evaluate z_i. The moving mesh equation (3.12) gives

$$\begin{aligned} &-\tau(\dot{x}_{i+1} - 2\dot{x}_i + \dot{x}_{i-1}) \\ &= \tfrac{1}{2}(u_{i+1}^{p-1} + u_i^{p-1})(x_{i+1} - x_i) - \tfrac{1}{2}(u_i^{p-1} + u_{i-1}^{p-1})(x_i - x_{i-1}), \end{aligned} \tag{4.36}$$

which has first integral

$$-\tau(\dot{x}_{i+1} - \dot{x}_i) = \frac{1}{2}(u_{i+1}^{p-1} + u_i^{p-1})(x_{i+1} - x_i) + \theta(t). \tag{4.37}$$

Substituting for z_i and w_i we have after some manipulation that

$$\begin{aligned} &\tfrac{\tau}{2}(z_{i+1} - z_i)\left(1 + O(\tfrac{1}{s})\right) \\ &= \tfrac{1}{2}(w_{i+1}^{p-1} + w_i^{p-1})(z_{i+1} - z_i) + \tfrac{(T-t)^{1/2}}{g(s)}\theta(t). \end{aligned} \tag{4.38}$$

Choosing θ appropriately and using (4.34), this is a centred difference discretization of (4.10), and hence z_i is a discrete approximation to $z(\xi)$ given in (4.9).

We conclude that the discrete form of the PDE coupled with MMPDE6 admits an approximately self-similar solution for which both the mesh x_i and the function u_i are consistent approximations to the mesh $x(\xi, t)$ and function $u(\xi, t)$ calculated in §4.1. Provided that this solution is (like its analytic counterpart) an attractor for a wide class of initial data, then the numerical method given will faithfully reproduce the dynamics of the blow-up peak for all times up to the blow-up time. In particular, the asymptotic estimates given in §4.1 will be reproduced in the numerical calculations. It is difficult to analyze analytically whether the solution is an attractor, and we do not attempt it here.

5. Numerical examples. We now present some numerical calculations for several blow-up problems using MMPDE4 and MMPDE6. These calculations support the analysis of §§3 and 4 and also illustrate the effect of smoothing the monitor function M. For these calculations we take $u_0(x) = 20 \sin(\pi x)$.

After spatial discretization, the resulting ODE systems are solved using the double precision version of the stiff ODE solver DASSL [Pet82]. The time integration uses the backward differentiation formulas (BDF), wherein an approximate Jacobian is computed by DASSL internally using finite differences. The relative and absolute local time-stepping error tolerances

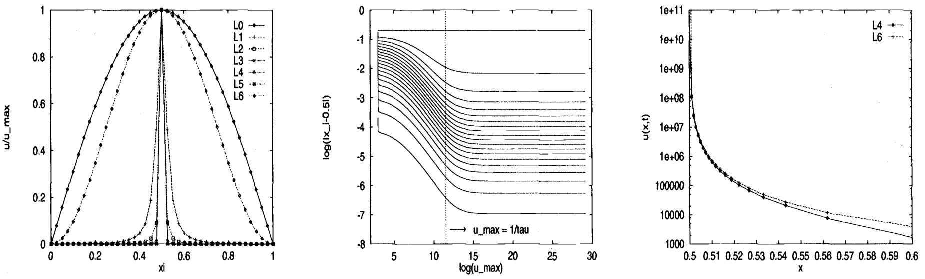


FIG. 1 (1a, 1b, *and* 1c *from the left*). *MMPDE4 without smoothing is used for solving* (2.1) *with* $p = 2$. *In Figs.* 1a *and* 1c $L0$: *initial solution*, $L1 : u_{\max} = 10^8$, $L2 : u_{\max} = 10^9$, $L3 : u_{\max} = 10^{10}$, $L4 : u_{\max} = 10^{11}$, $L5 : u_{\max} = 5 \times 10^{11}$, $L6$: *the asymptotic solution.*

(in a root-mean-square norm) are chosen as rtol $= 10^{-8}$ and atol $= 10^{-8}$, respectively. Unless stated otherwise, we use a uniform mesh (in x) initially with $N = 40$ and take $\tau = 10^{-5}$. As pointed out in [HRR94a] and [HRR94b], the choice of value for the time correction parameter τ is not critical and can generally be fixed to be a small positive value.

5.1. Calculations using MMPDE4. The analysis in §3 indicates that MMPDE4 ceases to evolve the mesh when the timescale of the blow-up is less than τ, i.e., when

$$(T - t) < \tau. \tag{5.1}$$

Since $u(x^*, t) \approx (T - t)^{-\beta}$, the mesh ceases to evolve when

$$u(x^*, t) > \tau^{-\beta}, \tag{5.2}$$

after which MMPDE4 gives a nonuniform but essentially *fixed* mesh. The results of [AB94] show that if $p > 2$, u_i will blow up at only one point and if $p = 2$ it will blow up at three points, but the blow-up is asymptotically dominated by the growth at one point.

To confirm these results we integrate (2.1) when $p = 2$ coupled with MMPDE4 using the monitor function given by (3.6). Blow-up occurs at $T = 0.082291$ and $x^* = \frac{1}{2}$ so that the maximum value occurs at u_{20}. In Fig. 1a we present a graph of the scaled discrete solution values $\{\frac{u_i}{u_{20}}\}$ for $u_{20} = 10^8$, 10^9, 10^{10}, 10^{11}, 5×10^{11}, and in Fig. 1b we show the corresponding mesh, plotting $\log |x_i - \frac{1}{2}|$ against $\log(u_{20})$. From the second figure it is clear that the mesh evolves until $u_{20} \approx \frac{1}{\tau}$ and is then fixed. The results of Fig. 1a show convergence of the normalized solution to a delta function, demonstrating that, effectively, it is only u_{20} which is blowing up. In Fig. 1c we plot the approximation for $u(x, t)$ when $u_{20} = 10^{11}$ and compare it with the asymptotic function $u(x, t)$ given in (2.3). It is clear that it approximates the limiting asymptotic function rather well (except at the blow-up point), showing that the mesh generated by MMPDE4 leads to an accurate approximation to $u(x, t)$ away from the point of blow-up and the behaviour of $u(x, T)$ is well approximated in this range. Indeed, at the point when MMPDE4 ceases to evolve the mesh, the mesh point nearest $\frac{1}{2}$ is given by

$$x_i = \frac{1}{2} + O(\Delta\xi \tau^{1/2} \log^{1/2} \tau), \tag{5.3}$$

where $\Delta\xi$ is the mesh spacing in the computational domain. Thus MMPDE4 gives an accurate picture of the evolution of the solution in the region $[0, \frac{1}{2} - O(\Delta\xi\tau^{1/2}\log^{1/2}\tau)] \cup [\frac{1}{2} + O(\Delta\xi\tau^{1/2}\log^{1/2}\tau), 1]$, but it will not resolve the structure in the remaining blow-up region.

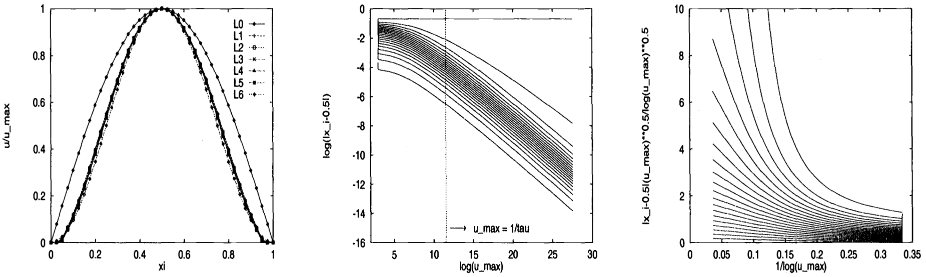


FIG. 2 (2a, 2b, *and* 2c *from the left*). *MMPDE*6 *without smoothing is used for solving* (2.1) *with* $p = 2$. *In Fig.* 2a *L*0: *initial solution*, $L1 : u_{\max} = 10^8$, $L2 : u_{\max} = 10^9$, $L3 : u_{\max} = 10^{10}$, $L4 : u_{\max} = 10^{11}$, $L5 : u_{\max} = 5\times 10^{11}$, *L*6: *the asymptotic solution.*

5.2. Calculations using MMPDE6. The analysis given in §§3 and 4 indicates that (in contrast to MMPDE4) using MMPDE6 with the monitor function given by (3.6) should give an accurate resolution of the blow-up peak. A calculation of the evolution of the PDE gives a blow-up time of $T = 0.082283$ when $p = 2$. The scaled solution $\{\frac{u_i}{u_{20}}\}$ for $u_{20} = 10^8,\ 10^9,\ 10^{10},\ 10^{11},\ 5\times 10^{11}$, is given in Fig. 2a together with the theoretical scaled solution $\cos^2\pi(\xi - \frac{1}{2})$ where $\xi = i\Delta\xi$ (cf. (2.10) and (4.17)). It is clear from the figure that there is close agreement between the predicted and computed curves implying that the solution calculated in §4.2 is indeed an attractor. Figure 2b shows a graph of $\log|x_i - \frac{1}{2}|$ as a function of $\log u_{\max}$ with clear evidence that each of the mesh points (apart from those close to the boundary) evolves in the same (approximately self-similar) manner. To confirm that the mesh behaves precisely as predicted in §4, we give a graph in Fig. 2c of $\frac{|x_i-\frac{1}{2}|}{(T-t)^{1/2}|\log(T-t)|^{1/2}}$ as a function of $|\log(T - t)|^{-1}$, where we use the estimate $T - t = \frac{1}{u_{20}}$. According to formula (4.16),

$$\frac{|x_i - \frac{1}{2}|}{(T - t)^{1/2}|\log(T - t)|^{1/2}} \approx \sqrt{8}\,\tan\pi\left(\xi_i - \frac{1}{2}\right)\left[1 - \frac{1}{2}\frac{\alpha}{|\log(T - t)|}\right] \tag{5.4}$$

as $|\log(T - t)| \to \infty$. Thus, for each i we expect to obtain a graph which is asymptotically linear in $|\log(T - t)|^{-1}$ and converges to $\sqrt{8}\,\tan\pi(\xi_i - \frac{1}{2})$ with slope $-\frac{1}{2}\alpha\sqrt{8}\,\tan\pi(\xi_i - \frac{1}{2})$. The figure clearly shows that this is the case, and we estimate α to be 7.

Thus MMPDE6 accurately reproduces the evolution of the peak profile; however, with 40 mesh points $x_{39} - \frac{1}{2} \approx 35.94(T - t)^{1/2}|\log(T - t)|^{1/2}$ so that $x_{39} = 0.500966$ when $u_{20} = 10^{11}$. Thus, the mesh points are concentrated in the blow-up region, so that the solution away from the peak is poorly resolved.

If smoothing as described in §3 is introduced, the resulting form of $\frac{u_i}{u_{20}}$ for the same value of $u_{\max}$ is given in Fig. 3a. In this case, the blow-up is slightly delayed to $T = 0.082319$. Now, the scaled discrete solution $\{\frac{u_i}{u_{20}}\}$ no longer approaches a constant curve, but instead, slowly approaches the delta function which would be obtained using a uniform mesh. This is because smoothing tends to place fewer points in the peak and more points away from the peak. Nevertheless, we see from Fig. 3b that the mesh points close to the peak evolve in a manner similar to those in Fig. 2b. Hence, whilst these points are close to the blow-up point, the analysis of §4 still applies and the value of u_i will be close to $u(x_i, t)$. Thus, although smoothing does not precisely align the mesh along the level curves of the function μ in (4.16), it still gives an accurate resolution of the structure of the peak.

The resolution of the peak in the two cases is made apparent in Figs. 4a and 4b, which plot $\frac{u_i}{u_{20}}$ as a function of $\log|x_i - \frac{1}{2}|$ for $u_{20} = 10^9,\ 10^{10}, 10^{11},\ 5\times 10^{11}$ for the unsmoothed

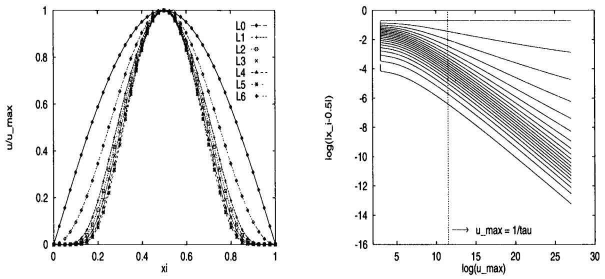


FIG. 3 (3a *and* 3b *from the left*). *MMPDE6 with smoothing* ($i_p = 2$) *is used for solving* (2.1) *with* $p = 2$. *In Fig.* 3a *L0: initial solution, L1* : $u_{\max} = 10^8$, *L2* : $u_{\max} = 10^9$, *L3* : $u_{\max} = 10^{10}$, *L4* : $u_{\max} = 10^{11}$, *L5* : $u_{\max} = 5 \times 10^{11}$, *L6: the asymptotic solution which is valid only for the unsmoothed case.*

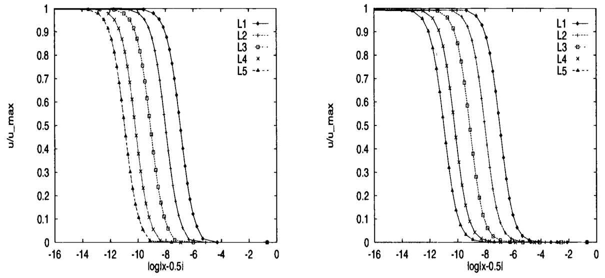


FIG. 4 (4a *and* 4b *from the left*). *MMPDE6 is used for solving* (2.1) *with* $p = 2$. *Fig.* 4a *is for the case without smoothing and Fig.* 4b *for that with smoothing* ($i_p = 2$). *In the figures, L1* : $u_{\max} = 10^8$, *L2* : $u_{\max} = 10^9$, *L3* : $u_{\max} = 10^{10}$, *L4* : $u_{\max} = 10^{11}$, *L5* : $u_{\max} = 5 \times 10^{11}$.

and smoothed meshes, respectively. The coordinate $\log |x_i - \frac{1}{2}|$ is used to stretch the length scale close to the peak so that its structure can be seen more easily. The two figures are almost identical, confirming that smoothing does not affect the accuracy of the resolution of the peak for these values of u (although presumably it would for much larger values). As a further test of the accuracy of the schemes the graphs in Figs. 4a and 4b can be compared with the graph of the function

$$\frac{u}{u_{\max}} = \left[1 + \frac{(x - \frac{1}{2})^2}{8(T - t)(\alpha - \log(T - t))}\right]^{-1} \tag{5.5}$$

where $u_{\max} \equiv (T - t)^{-1}$. Indeed, the value of $\alpha = 7$ estimated earlier gives a good fit for each of the curves.

Although smoothing is not necessary when $p = 2$, it becomes essential when dealing with more severe nonlinearities, as unsmoothed moving mesh equations can tend to introduce oscillations in solutions away from singularities by placing too few points there (e.g., see [VBFZ89]). We see this by taking $p = 5$, $M = u^4$, and using an initial mesh equidistributed with respect to the initial solution $u(x, 0) = 20 \sin(\pi x)$. Figure 5a shows the computed value of $\left(\frac{u_i}{u_{\max}}\right)^4$ plotted as a function of ξ in the case of no smoothing for $u_{\max}^5 = 10^8, 10^9, 10^{10}, 10^{11}$, and 10^{12}. These computed solutions approximate $\cos^2 \pi(\xi - \frac{1}{2})$ for $u_{\max}^5 = 10^{11}$, but deviate from it, due to mesh oscillations, when $u_{\max}^5 > 10^{11}$. In contrast, the smoothed solution (obtained with $i_p = 4$) does not oscillate and still retains enough points near the blow-up point to give good resolution in the peak, where for reference purposes we also give the asymptotic solution $\cos^2 \pi(\xi - \frac{1}{2})$ which applies only for the nonsmooth moving mesh case (see Fig. 5b).

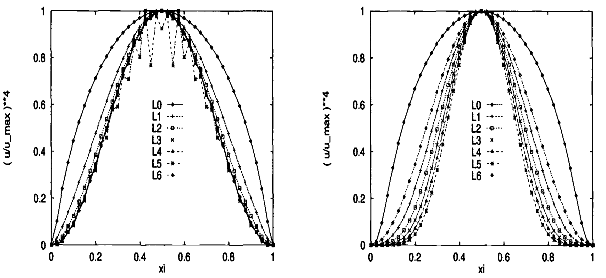


FIG. 5 (5a *and* 5b *from the left*). *MMPDE6 is used for solving* (2.1) *with* $p = 5$. *In Fig.* 5a (*without smoothing*), *L0: initial solution,* $L1 : u^5_{\max} = 10^8$, $L2 : u^5_{\max} = 10^9$, $L3 : u^5_{\max} = 10^{10}$, $L4 : u^5_{\max} = 10^{11}$, $L5 : u^5_{\max} = 10^{12}$, *L6: the asymptotic solution. In Fig.* 5b (*with smoothing* ($i_p = 4$)), *L0: initial solution,* $L1 : u^5_{\max} = 10^9$, $L2 : u^5_{\max} = 10^{10}$, $L3 : u^5_{\max} = 10^{11}$, $L4 : u^5_{\max} = 10^{12}$, $L5 : u^5_{\max} = 10^{13}$, *L6: the asymptotic solution which is valid only for the unsmoothed case.*

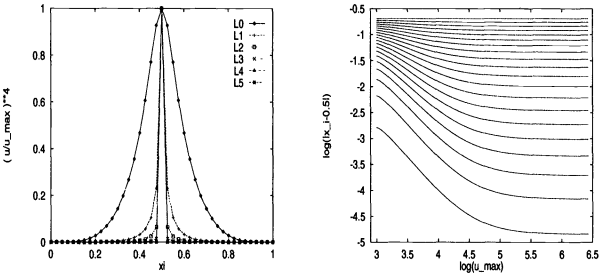


FIG. 6 (6a *and* 6b *from the left*). *MMPDE6 with smoothing* ($i_p = 4$) *and the arclength monitor function is used for solving* (2.1) *with* $p = 5$. *In Fig.* 6a, *L0: initial solution,* $L1 : u^5_{\max} = 10^{10}$, $L2 : u^5_{\max} = 10^{11}$, $L3 : u^5_{\max} = 10^{12}$, $L4 : u^5_{\max} = 10^{13}$, $L5 : u^5_{\max} = 10^{14}$, *L6: the asymptotic solution.*

It is interesting to repeat the calculations for $p = 5$ using the common choice of the arclength monitor function $M = \sqrt{1 + u_x^2}$ [HRR94b]. The results of §4 indicate that the mesh will cease to evolve when $\tau u_{\max}^{(p-3)/2} >> 1$, which in this case occurs when $u_{\max} >> 1/\tau$. In Figs. 6a and 6b we present the resulting computed solution for different values of $u^5_{\max}$ and the evolution of the mesh (with smoothing $i_p = 4$), respectively. We see clearly that the mesh ceases to evolve in this case. The value of $u_{\max}$ at which this occurs is, in fact, rather smaller than the value predicted above. This is a result of the smoothing we have employed.

5.3. Exponential nonlinearities. We can easily extend the results of the previous subsections to the case of an exponential nonlinearity, such as for the equation

$$u_t = u_{xx} + e^u \tag{5.6}$$

and its generalization

$$u_t = (|u_x|^\sigma u_x)_x + e^u, \tag{5.7}$$

using the monitor function $M(u) = e^u$. It is well known [Dol85] that close to the blow-up point the solutions of (5.6) have an approximately self-similar solution of the form

$$u(x,t) \sim -\log(T-t) - \log\left(1 + \frac{x^2}{4(T-t)\log(T-t)}\right). \tag{5.8}$$

The solutions of (5.7) are, in contrast, self-similar such that

$$u(x,t) \sim -\log(T-t) + W\left(\frac{x}{(T-t)^{1/(\sigma+2)}}\right), \tag{5.9}$$

where for small σ the function W has the form

$$W(\zeta) = -\log\left(1 + \frac{1}{2}\left(\frac{\sigma}{2}\right)^{1-\sigma}\zeta^2\right) \tag{5.10}$$

(see [BG94]). Using an analysis identical to that given earlier we may derive the following formal result.

PROPOSITION 5.1. *Let $M(u) = e^u$.*

(i) *If $\sigma = 0$, then an equidistributed mesh transformation is given by*

$$x(\xi,t) = x^* + (4(T-t)\log(T-t))^{1/2}\tan^{-1}\left(\pi\left(\xi - \frac{1}{2}\right)\right), \tag{5.11}$$

$$\frac{e^{u(\xi,t)}}{e^{u(\frac{1}{2},t)}} \to \cos^2\pi\left(\xi - \frac{1}{2}\right). \tag{5.12}$$

(ii) *Let*

$$f(x) \equiv \int_0^x \frac{dx}{1+y^{\sigma+2}}, \quad F \equiv 2f(\infty). \tag{5.13}$$

If $\sigma > 0$, σ small, then

$$x(\xi,t) = x^* + \left(2(T-t)\left(\frac{2}{\sigma}\right)^{1-\sigma}\right)^{1/(\sigma+2)} f^{-1}\left[F\left(\xi - \frac{1}{2}\right)\right], \tag{5.14}$$

$$\frac{e^{u(\xi,t)}}{e^{u(\frac{1}{2},t)}} \to \frac{1}{1+\left(f^{-1}\left[F(\xi - \frac{1}{2})\right]\right)^2}. \tag{5.15}$$

Computations with the exponential nonlinearity raise issues similar to those discussed earlier, especially with regard to smoothing. For example, if $\sigma = 0$ then a calculation with $u_0(x) = 5\sin(\pi x)$ and no smoothing is successful when $u_{\max} < 15$ ($e^{u_{\max}} < 3 \times 10^6$), giving a close approximation to the asymptotic results, but it becomes unstable for larger values of u. We illustrate this in Fig. 7a, giving $e^{u_i - u_{\max}}$ for $u_{\max} = 10, 11.6, 13.33, 15, 16.6$. A graph of the resulting mesh in Fig. 7b shows the oscillations in the mesh points at the boundary even more dramatically. In Figs. 7c and 7d we give the corresponding results with smoothing, taking $i_p = 4$.

It is interesting to compare the functions $u(x,t)$ computed in these cases, and in Figs. 7e and 7f we present graphs of $e^{u-u_{\max}}$ as a function of the physical coordinate x for $u_{\max} = 10, 11.6, 13.33, 15, 16.6$ for both the unsmoothed and smoothed cases, respectively. These graphs are nearly identical, except that the peak is smoother in Fig. 7f. Figure 8 gives the results of the same calculation on a fixed, uniform mesh. We conclude again that smoothing greatly improves the method without losing the advantages of mesh adaption.

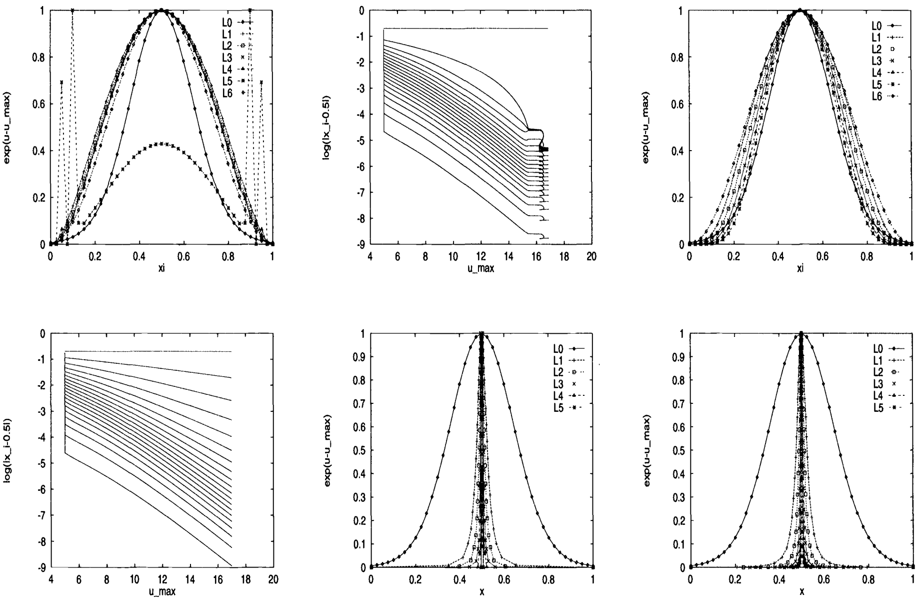


FIG. 7 (7a, 7b, 7c, 7d, 7e, *and* 7f *from the top left*). *MMPDE*6 *is used for solving* (5.6). *Figures* 7a, 7b, *and* 7e *are for the case without smoothing and Figs.* 7c, 7d, *and* 7f *are for that with smoothing* ($i_p = 4$). *In the figures, L*0*: initial solution,* $L1 : u_{\max} = 10$, $L2 : u_{\max} = 11.6$, $L3 : u_{\max} = 13.33$, $L4 : u_{\max} = 15$, $L5 : u_{\max} = 16.6$, *L*6*: the asymptotic solution which is valid only for the unsmoothed case.*

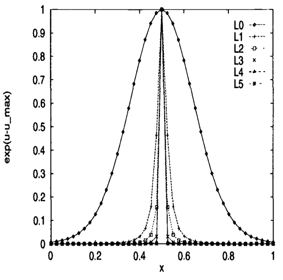


FIG. 8. *A fixed, uniform mesh is used for solving* (5.6). *L*0*: initial solution,* $L1 : u_{\max} = 10$, $L2 : u_{\max} = 11.6$, $L3 : u_{\max} = 13.33$, $L4 : u_{\max} = 15$, $L5 : u_{\max} = 16.6$.

5.4. Degenerate parabolic equations. To show the ability of the MMPDE methods to handle different types of blow-up, we conclude this section with the application of the MMPDE methods to a problem that has not been as extensively analyzed as (2.1) or (2.20). We consider the degenerate parabolic problem

$$xu_t = u_{xx} + 15u^2 \tag{5.16}$$

subject to the boundary conditions $u(0, t) = u(1, t) = 0$. This problem has been studied theoretically and numerically in [Flo91] and [SF90]. It is shown in [Flo91] that a distinct feature of the blow-up for (5.16) from that for nondegenerate problems (e.g., (1.1)) is that the solution blows up at the boundary $x = 0$. In contrast to that for nondegenerate problems, the blow-up behaviour for degenerate problems like (5.16) is as yet not well understood.

The results obtained with MMPDE6 and smoothing ($i_p = 2$) are shown in Figs. 9a–9e. In the computation, we have used $u_0(x) = 4x(1-x)$ and $M = u^{3/2}$, which is chosen so that the

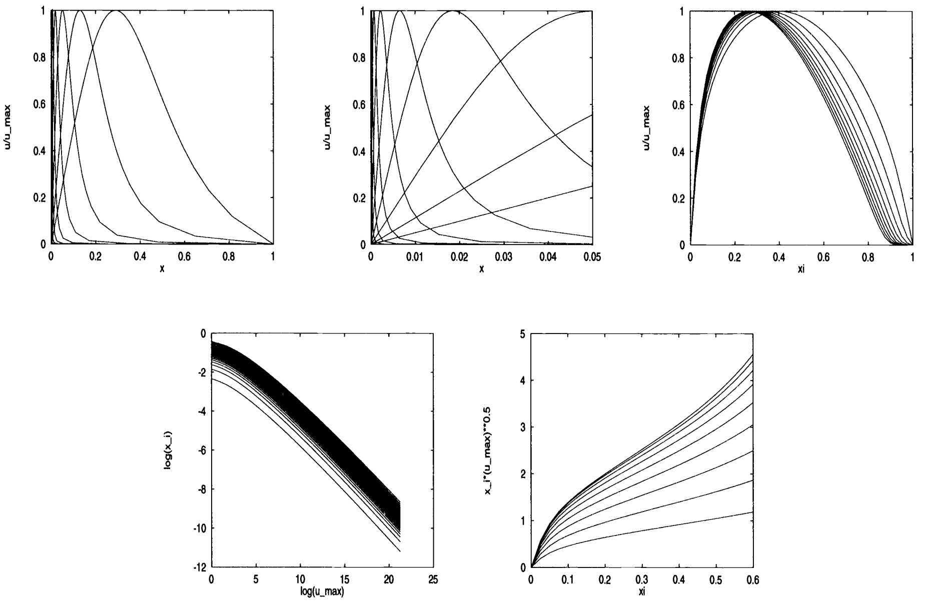


FIG. 9 (9a, 9b, 9c, 9d, *and* 9e *from the top left*). *MMPDE*6 *with smoothing* ($i_p = 2$) *is used for solving* (5.16). *Figure* 9b *is the magnification of the left portion of Fig.* 9a.

extended system consisting of the physical PDE and the MMPDE preserves the underlying scaling invariance of (5.16). The results show the blow-up at the boundary $x = 0$. Our computation gives a blow-up time of $T \approx 0.056015$, which is close to the value obtained in [SF90] with a so-called peak-tracking strategy. The computation required roughly 36.7 seconds of CPU time on an SGI R3000 Indigo.

6. Conclusions and comments. In this paper we have considered PDEs which model blow-up problems for which scaling invariance plays a natural role in describing the underlying solution structures. When one computes the solutions to such problems, adaptive mesh methods are virtually unavoidable. It is useful to interpret such numerical methods as discretizations (on a uniform mesh) of the PDEs rewritten in terms of a computational coordinate transformation. This transformation can in turn be defined through an MMPDE [HRR94b] which is determined by equidistribution with respect to a monitor function M. It is natural to seek a monitor function which preserves the scaling invariance; note that this does not require detailed knowledge of the solution behaviour itself. The scaling invariance is then preserved by the discretization, i.e., by the actual moving mesh method which is implemented. To our knowledge, this is the first instance in which rigorous analysis has been used to motivate the choice of specific monitor functions. For the blow-up problems with known detailed solution behaviour, the solutions are only approximate similarity solutions, and, quite remarkably, their structure is also preserved by the discrete equations when suitable monitor functions are chosen. As a result, with relatively few mesh points the analytic structure of the blow-up solutions can be accurately computed in a very efficient way. We briefly present some computations (without accompanying analysis) for a degenerate parabolic problem whose solution has not been extensively analyzed.

A comparison of our approach for blow-up problems with those of others is outside the scope of this paper. Nevertheless, we have tried to emphasize the naturalness of our method for automatically picking up the self-similar coordinate and to demonstrate the ease with which these problems can often be solved with little a priori knowledge.

This is not to say that the moving mesh methods are without pitfalls, however. It is important to use proper monitor functions, and examples are given where the popular choice of arclength fails to perform adequately. The MMPDEs must also be implemented with care. For example, in Figs. 1a and 1b MMPDE4 with fixed τ fails to give the proper blow-up structure when the solution becomes sufficiently large, and the question of whether or not to use mesh smoothing, and if so, the judicious choice of one, are issues with no simple resolution. The blow-up problems considered here are part of a larger framework of PDEs having similarity solutions for which MMPDEs with suitable monitor functions preserve the scaling invariance, and we are in the process of developing a theory for such problems.

REFERENCES

[AB94] L. Abia and C. J. Budd, *Blow-up in uniform discretisations of parabolic PDEs*, preprint, 1994.

[BB92] J. Bebernes and S. Bricher, *Final time blowup profiles for semilinear parabolic equations via centre manifold theory*, SIAM J. Appl. Math., 23 (1992), pp. 852-869.

[BDG93] C. J. Budd, J. W. Dold, and V. A. Galaktionov, *Self-similar solutions of a quasilinear diffusion problem*, submitted for publication.

[BDS93] C. J. Budd, J. W. Dold, and A. M. Stuart, *Blowup in a parabolic equation with constraints*, SIAM J. Appl. Math., 53 (1993), pp. 718-742.

[BE89] J. Bebernes and D. Eberly, *Mathematical Problems for Combustion Theory*, Appl. Math. Sci., 83, Springer-Verlag, 1989.

[Ber89] M. Berger, *Some applications of adaptive mesh refinement*, Adaptive Methods for Partial Differential Equations, J. E. Flaherty, P. J. Paslow, M. S. Shephard, and J. D. Vasilakis, eds., SIAM, Philadelphia, 1989.

[BG94] C. J. Budd and V. A. Galaktionov, *Critical diffusion exponents for blow-up in quasi-linear elliptic PDEs with exponential source*, Proc. Roy. Soc. Edinb., to appear.

[BK88] M. Berger and R. V. Kohn, *A rescaling algorithm for the numerical calculation of blowing-up solutions*, Comm. Pure Appl. Math., 41 (1988), pp. 841-863.

[Cho81] A. J. Chorin, *Estimates of intermittency, spectra, and blow-up in developed turbulence*, Comm. Pure Appl. Math., 34 (1981), pp. 853-866.

[DD87] E. A. Dorfi and L. O'c. Drury, *Simple adaptive grids for 1-D initial value problems*, J. Comp. Phys., 69 (1987), pp. 175-195.

[Dol85] J. W. Dold, *Analysis of the early stage of thermal runaway*, Quart. J. Mech. Appl. Math., 38 (1985), pp. 361-387.

[FK92] S. Filippas and R. Kohn, *Refined asymptotics for the blow up of* $u_t - \Delta u = u^p$, Comm. Pure Appl. Math., 45 (1992), pp. 821-869.

[Flo91] M. S. Floater, *Blow-up at the boundary for degenerate semilinear parabolic equations*, Arch. Rational Mech. Anal., 114 (1991), pp. 57-77.

[FM85] A. Friedman and B. Mcleod, *Blow up of positive solutions of semilinear heat equations*, Indiana Univ. Math. J., 34 (1985), pp. 425-447.

[FVZ90] R. M. Furzeland, J. G. Verwer, and P. A. Zegeling, *A numerical study of three moving grid methods for one-dimensional partial differential equations which are based on the method of lines*, J. Comp. Phys., 89 (1990), pp. 349-388.

[Gel63] I. M. Gelfand, *Some problems in the theory of quasilinear equations*, Amer. Math. Soc. Trans., 29 (1963), pp. 295-381.

[GK85] Y. Giga and R. Kohn, *Asymptotically self-similar blowup of semilinear heat equations*, Comm. Pure Appl. Math., 38 (1985), pp. 297-319.

[GP91] V. A. Galaktionov and S. A. Posashkov, *Single point blow-up for N-dimensional quasilinear equations with gradient diffusion and source*, Indiana Univ. Math. J., 40 (1991), pp. 1041-1060.

[GV93] V. A. Galaktionov and J. L. Vasquez, *Blow-up for quasilinear heat equations described by means of nonlinear Hamilton-Jacobi equations*, preprint 1993.

[HRR94a] W. Huang, Y. Ren, and R. D. Russell, *Moving mesh partial differential equations* (*MMPDEs*) *based on the equidistribution principle*, SIAM J. Numer. Anal., 31 (1994), pp. 709-730.

[HRR94b] ———, *Moving mesh methods based upon moving mesh partial differential equations*, J. Comp. Phys., 113 (1994), pp. 279-290.

[HV93] M. Herrero and J. Velazquez, *Blow-up behaviour for one-dimensional semilinear parabolic equations*, Ann. Inst. H. Poincaré, Analyse nonlinéaire, 10 (1993), pp. 131-189.

[HL89] J. M. Hyman and B. Larrouturou, *Dynamic rezone methods for partial differential equations in one space dimension*, Appl. Numer. Math., 5 (1989), pp. 435-450.

[Kas77] D. Kassoy, *The supercritical spatially homogeneous thermal explosion: Initiation to completion*, Quart. J. Mech. Appl. Math., 30 (1977), pp. 71-89.

[LPSS86] B. LeMesurier, G. Papanicolaou, C. Sulem, and P.-L. Sulem, *The focusing singularity of the nonlinear Schrödinger equation*, Phys. Rev. A, 34 (1986), pp. 1200-1210.

[Pet82] L. R. Petzold, *A description of DASSL: A differential/algebraic system solver*, SAND82-8637, Sandia Labs., Livermore, CA, 1982.

[SF90] A. M. Stuart and M. S. Floater, *On the computation of blow-up*, Euro. J. Appl. Math., 1 (1990), pp. 47-71.

[VBFZ89] J. G. Verwer, J. G. Blom, R. M. Furzeland, and P. A. Zegeling, *A moving grid method for one-dimensional PDEs based on the method of lines*, Adaptive Methods for Partial Differential Equations, J. E. Flaherty, P. J. Paslow, M. S. Shephard, and J. D. Vasilakis, eds., SIAM, Philadelphia, 1989.

SIAM J. SCI. COMPUT.
Vol. 17, No. 2, pp. 328-346, March 1996

002

HIGH-ACCURACY FINITE-DIFFERENCE SCHEMES FOR LINEAR WAVE PROPAGATION*

DAVID W. ZINGG†, HARVARD LOMAX‡, AND HENRY JURGENS†

Abstract. Two high-accuracy finite-difference schemes for simulating long-range linear wave propagation are presented: a maximum-order scheme and an optimized scheme. The schemes combine a seven-point spatial operator and an explicit six-stage low-storage time-march method of Runge-Kutta type. The maximum-order scheme can accurately simulate the propagation of waves over distances greater than five hundred wavelengths with a grid resolution of less than twenty points per wavelength. The optimized scheme is found by minimizing the maximum phase and amplitude errors for waves which are resolved with at least ten points per wavelength, based on Fourier error analysis. It is intended for simulations in which waves travel under three hundred wavelengths. For such cases, good accuracy is obtained with roughly ten points per wavelength.

Key words. finite-difference schemes, wave propagation, phase error, high accuracy

AMS subject classifications. 65M05, 76-08, 78-08

1. Introduction.

In the past few years, interest has grown in time-domain numerical simulations of linear wave phenomena using finite-difference or related methods. Applications include electromagnetic [12, 13, 20], acoustic [3, 6, 14], and elastic waves [10, 11]. It is generally recognized that in order to avoid excessively fine meshes for many practical problems, high-order discretizations are required. Consequently, many high-order differencing methods have been developed for problems involving wave phenomena [1, 4, 5, 7, 10].

Furthermore, optimized or spectral-like finite-difference schemes have been proposed which can provide improvements in accuracy over high-order schemes with the same computational effort [8, 9]. In an optimized finite-difference scheme, the error behavior over a range of spatial wavenumbers is optimized according to some criterion, usually based on Fourier error analysis. This contrasts with conventional schemes, which generally maximize the order of accuracy, i.e., the order of the leading error term. Detailed studies of optimized schemes have been performed by Lele [9] and Holberg [8], who optimized the spatial operator only. Sguazzero, Kindelan, and Kamel [11] have developed optimized fully discrete schemes based on Holberg's spatial operators. Tam and Webb [15] present an optimized scheme which consists of a seven-point centered-difference operator in space combined with a four-step time-marching method of the Adams-Bashforth type.

In this paper, we present two fully discrete finite-difference schemes for linear wave propagation: a maximum-order scheme and an optimized scheme. The schemes are suitable for problems requiring high accuracy with relatively large distances of travel. Both schemes combine a seven-point spatial operator and an explicit six-stage time-march method of the Runge-Kutta type. The spatial operator is divided into an antisymmetric component, i.e., a centered difference operator, and a symmetric component or filter. The optimized scheme is developed by minimizing the maximum phase and amplitude errors obtained using Fourier error analysis for waves which are resolved with at least ten points per wavelength.

The maximum-order scheme and the optimized scheme are presented in the next two sections. The two schemes are then analyzed and compared. Next, their stability is discussed and numerical boundary schemes are presented. After presenting the results of some numerical experiments, we conclude with a discussion of some of the considerations involved in choosing a difference scheme for a given problem.

*Received by the editors March 4, 1994; accepted for publication (in revised form) October 24, 1994.

†Institute for Aerospace Studies, University of Toronto, 4925 Dufferin Street, Downsview, Ontario, Canada M3H 5T6 (dwz@oddjob.utias.utoronto.ca, jurgens@oddjob.utias.utoronto.ca).

‡NASA Ames Research Center, Moffett Field, CA 94035 (lomax@nas.nasa.gov).

2. Maximum-order scheme. We consider first the spatial difference operator, which is divided into an antisymmetric component, i.e., a centered difference operator, and a symmetric component, or filter, which provides dissipation of high wavenumber modes. The grid is uniform with $x_j = j\Delta x$. The function values at the grid nodes are $u_j = u(x_j)$. For the linear convection equation with a positive phase speed, the first derivative of u at node j is approximated as

$$(1)\quad \begin{aligned}(\delta_x u)_j = \frac{1}{\Delta x}[&(d_3 - a_3)u_{j-3} + (d_2 - a_2)u_{j-2} + (d_1 - a_1)u_{j-1} + d_0 u_j \\ &+ (d_1 + a_1)u_{j+1} + (d_2 + a_2)u_{j+2} + (d_3 + a_3)u_{j+3}],\end{aligned}$$

where the a_i are the coefficients of the antisymmetric component and the d_i are the coefficients of the filter. The maximum order of accuracy possible for this operator is sixth-order. This is obtained with $a_1 = 3/4$, $a_2 = -3/20$, $a_3 = 1/60$, $d_0 = d_1 = d_2 = d_3 = 0$. With a nonzero value of d_0, fifth-order accuracy is obtained with $d_1 = -3d_0/4$, $d_2 = 3d_0/10$, $d_3 = -d_0/20$.

Since the time-marching method described below is unstable for pure imaginary eigenvalues, a nonzero value of d_0 is required. We have chosen $d_0 = 0.1$. Although this introduces some amplitude error, the amplitude error of the fully discrete scheme is generally less than the phase error, as will be shown later. Furthermore, the resulting operator produces dissipation for high wavenumber components of the solution which are not propagated accurately.

In order to apply this spatial operator to a hyperbolic system of equations in the form

$$(2)\quad \frac{\partial \mathbf{u}}{\partial t} + \frac{\partial(\mathbf{Au})}{\partial x} = 0,$$

the operator must be split into the antisymmetric part, δ_x^a (with the a_i coefficients), and the symmetric part, δ_x^s (with the d_i coefficients). The spatial derivative in equation (2) is approximated as

$$(3)\quad \frac{\partial(\mathbf{Au})}{\partial x} \approx \delta_x^a \mathbf{Au} + \delta_x^s |\mathbf{A}|\mathbf{u}$$

where

$$|\mathbf{A}| = \mathbf{X}|\Lambda|\mathbf{X}^{-1}$$

and $\mathbf{X}$ is the matrix of right eigenvectors and Λ the matrix of eigenvalues of $\mathbf{A}$.

The time-marching method is an explicit six-stage method of the following form:

$$(4)\quad \begin{aligned} u_{n+\alpha_1}^{(1)} &= u_n + h\alpha_1 f_n, \\ u_{n+\alpha_2}^{(2)} &= u_n + h\alpha_2 f_{n+\alpha_1}^{(1)}, \\ u_{n+\alpha_3}^{(3)} &= u_n + h\alpha_3 f_{n+\alpha_2}^{(2)}, \\ u_{n+\alpha_4}^{(4)} &= u_n + h\alpha_4 f_{n+\alpha_3}^{(3)}, \\ u_{n+\alpha_5}^{(5)} &= u_n + h\alpha_5 f_{n+\alpha_4}^{(4)}, \\ u_{n+1} &= u_n + h f_{n+\alpha_5}^{(5)}, \end{aligned}$$

where $u_n = u(t_n)$, h is the time step, and

$$f^{(k)} = \frac{du^{(k)}}{dt}.$$

With $\alpha_5 = 1/2$, this method is second-order accurate. With $\alpha_5 = 1/2, \alpha_4 = 1/3, \alpha_3 = 1/4, \alpha_2 = 1/5, \alpha_1 = 1/6$, it produces sixth-order accuracy for linear homogeneous ordinary differential equations. In [19], this six-stage method is shown to be more accurate than the classical fourth-order Runge-Kutta method when combined with the above spatial operator for both homogeneous and inhomogeneous linear problems. The extra stages were accounted for by performing the comparison for an equal number of derivative function evaluations. Furthermore, the six-stage method requires less computer memory than the classical fourth-order Runge-Kutta method.

3. Optimized scheme. Optimized finite-difference schemes are obtained by dropping the requirement of maximum order of accuracy and selecting the resulting free parameters to achieve some desired error behavior. The spatial operator given in (1) is at least first-order accurate if

$$d_0 + 2d_1 + 2d_2 + 2d_3 = 0 \tag{5}$$

and

$$a_1 + 2a_2 + 3a_3 = \frac{1}{2}. \tag{6}$$

A value of $d_0 = 0.1$ was selected based on stability considerations and the need for high wavenumber damping, as discussed above. Therefore, by reducing the order of accuracy of the spatial operator to first order, four free parameters are obtained. The time-march method, (4), is at least second-order accurate as long as $\alpha_5 = 0.5$. With this constraint, four free parameters are available in the time-march method as well. Consequently, eight parameters are available to optimize the fully discrete scheme.

The present optimized scheme was developed to minimize the maximum error for waves resolved with at least ten grid points per wavelength (*PPW*) for Courant numbers less than or equal to one. There are numerous ways to find such an optimized scheme. We used the following approach. First, we determined the optimal spatial operator for the specified wavenumber range in one dimension. Using this spatial operator, we then optimized the time-march method at a Courant number of one, also in one dimension. The resulting coefficients are listed below:

$$\begin{gathered} a_1 = 0.7599613, \quad a_2 = -0.1581220, \quad a_3 = 0.01876090, \\ d_0 = 0.1, \quad d_1 = -0.07638461, \quad d_2 = 0.03228961, \quad d_3 = -0.005904994, \\ \alpha_1 = 0.168850, \quad \alpha_2 = 0.197348, \quad \alpha_3 = 0.250038, \quad \alpha_4 = 0.333306, \quad \alpha_5 = 0.5. \end{gathered} \tag{7}$$

4. Fourier error analysis. In order to analyze finite-difference schemes for wave propagation problems, we consider the linear convection equation [16], given in one dimension by

$$\frac{\partial U}{\partial t} + c\frac{\partial U}{\partial x} = 0, \tag{8}$$

where $U = U(x,t)$ and c, the phase speed, is a positive real constant. With an initial condition given by

$$U(x,0) = U_0 \exp(i\kappa x), \tag{9}$$

the exact solution on an infinite domain is

$$U(x,t) = U_0 \exp[i\kappa(x - ct)], \tag{10}$$

where κ is the spatial wavenumber. Assuming a solution in the form

$$U(x,t) = u(t)\exp(i\kappa x) \tag{11}$$

and replacing the spatial derivative appearing in (8) by a numerical approximation, the following ordinary differential equation is obtained:

$$\frac{du}{dt} = -ic\kappa^* u = \lambda u, \tag{12}$$

where κ^* is the numerical (or modified) wavenumber, which depends on the spatial operator, and $\lambda = -ic\kappa^*$.

This ordinary differential equation can be numerically advanced in time using a time-march method. For linear time-march methods, the characteristic polynomial has one or more roots, σ, which are functions of λh, where h is the time step. We consider here only methods which produce one σ root. The numerical solution to (12) is then

$$u_n = U_0\sigma^n, \tag{13}$$

where $u_n = u(t_n) = u(nh)$. Writing $\sigma = R\exp(i\phi)$, the numerical solution to (8) is thus

$$U_{\text{num}}(x,t) = U_0R^n\exp[i(\kappa x + n\phi)]. \tag{14}$$

This numerical solution can differ from the exact solution in both amplitude and phase. We can rewrite the numerical solution as

$$U_{\text{num}}(x,t) = U_0R^n\exp[i\kappa(x - c^*t)], \tag{15}$$

where c^* is the numerical phase speed given by $c^* = \phi/\kappa h$. Comparing (15) with (10), we define the normalized error components as

$$er_a = |\sigma| - 1 = R - 1, \tag{16}$$

$$er_p = \frac{c^*}{c} - 1 = -\frac{\phi}{c\kappa h} - 1, \tag{17}$$

where er_a and er_p denote amplitude and phase error, respectively.

In one dimension, the error resulting from a given numerical scheme depends on the product $z = \kappa\Delta x$ where Δx is the grid spacing and the Courant number is $C = ch/\Delta x$. In two or three dimensions, the error has a further dependence on the direction of propagation. The two-dimensional linear convection equation is given by

$$\frac{\partial U}{\partial t} + c\cos\theta\frac{\partial U}{\partial x} + c\sin\theta\frac{\partial U}{\partial y} = 0. \tag{18}$$

This equation governs a plane wave convecting a scalar U with speed c along a straight line making an angle θ with respect to the x-axis. On a square grid, when the same difference operator is used to approximate the two spatial derivatives in (8), the two-dimensional numerical wavenumber κ^*_{2d} can be written as

$$\kappa^*_{2d}(z,\theta) = \cos\theta\kappa^*_{1d}(z\cos\theta) + \sin\theta\kappa^*_{1d}(z\sin\theta), \tag{19}$$

where κ^*_{1d} is determined from the one-dimensional analysis.

For the spatial operator given in (1), the numerical wavenumber is given by

$$
\begin{aligned}
i\kappa^* = \frac{1}{\Delta x}[d_0 + 2(d_1 \cos z + d_2 \cos 2z + d_3 \cos 3z) \\
+ 2i(a_1 \sin z + a_2 \sin 2z + a_3 \sin 3z)].
\end{aligned} \tag{20}
$$

For the time-marching method (4), σ is given by

$$
\sigma = 1 + \lambda h + \beta_2(\lambda h)^2 + \beta_3(\lambda h)^3 + \beta_4(\lambda h)^4 + \beta_5(\lambda h)^5 + \beta_6(\lambda h)^6, \tag{21}
$$

where

$$
\begin{gathered}
\beta_2 = \alpha_5, \quad \beta_3 = \alpha_5\alpha_4, \quad \beta_4 = \alpha_5\alpha_4\alpha_3, \\
\beta_5 = \alpha_5\alpha_4\alpha_3\alpha_2, \quad \beta_6 = \alpha_5\alpha_4\alpha_3\alpha_2\alpha_1.
\end{gathered}
$$

The procedure for determining the errors for given values of z, θ, and C proceeds as follows. First, κ^*_{2d} must be calculated from (19) with κ^*_{1d} determined from (20). The parameter λ is then found using $\lambda = -ic\kappa^*$. Equation (21) is used to determine σ, which then produces the phase and amplitude errors from (16) and (17).

Figures 1 and 2 show the numerical phase speed and amplitude at a Courant number of unity in one dimension for the following three fully discrete finite-difference schemes: (1) second-order centered differences in space with fourth-order Runge-Kutta time marching (designated RK4C2 on the figures), (2) fourth-order centered differences with fourth-order Runge-Kutta time marching (designated RK4C4) and (3) the maximum-order spatial scheme given by (1) with $d_0 = 0.1$ and the time-marching method given by (4) with the values of the coefficients which produce the maximum order (designated maximum-order). Care must be taken in assessing schemes at individual values of θ and C. However, for the schemes considered here, the errors in the practical range of wavenumbers are generally largest for $\theta = 0^o$ and decrease with decreasing Courant number.

Lele [9] defines the resolving efficiency of a scheme as the fraction of the domain $0 \leq z \leq \pi$ for which the errors lie below a specified tolerance. Table 1 shows the resolving efficiency of the three schemes above for four different values of error tolerance e. In all three cases, the phase error is larger than the amplitude error and hence the resolving efficiency is determined by the phase error. Note that as the error tolerance decreases, the higher-order schemes become relatively more efficient. For example, at the largest error tolerance shown, the resolving efficiency of the maximum-order scheme is less than twice that of scheme RK4C4 while, at the smallest tolerance, the maximum-order scheme has almost three times the resolving efficiency of scheme RK4C4. The parameter $z = \kappa\Delta x$ is related to the number of points per wavelength by which a given wave is resolved through the relation $PPW = 2\pi/z$. Table 2 shows the PPW required to produce errors below the specified error tolerance values.

The procedure used to develop the optimized scheme is not sufficient to guarantee that the maximum error obtained over the specified wavenumber range in one dimension is not exceeded at nonzero values of θ. However, in the present case, the maximum phase and amplitude errors are obtained at $\theta = 0^o$ at a Courant number of unity. These errors are shown in Figures 3 and 4 in comparison with the maximum-order scheme for $0 \leq z \leq \pi/5$. The errors produced by the optimized scheme are bounded by 2×10^{-5} while the maximum-order scheme produces errors up to 4×10^{-4}. For the range $0.4 \leq z \leq 0.7$, the optimized scheme gives much smaller errors than the maximum-order scheme. This corresponds roughly to 9 to 16 PPW. For larger wavenumbers the advantage is reduced, while for smaller wavenumbers, the maximum-order scheme is more accurate.

Table 3 shows the resolving efficiency r and required PPW values for the optimized scheme with the four error tolerances as in the preceding tables. The advantage of the optimized

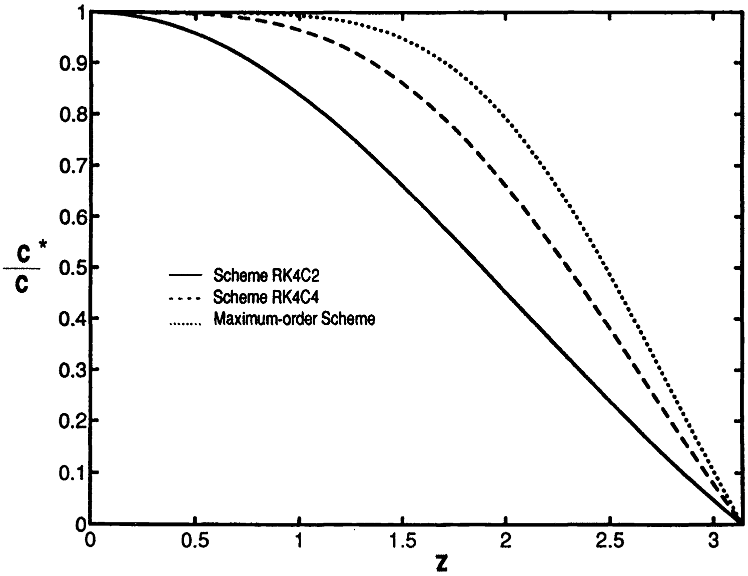


FIG. 1. *Numerical phase speed for second-order centered differences with fourth-order Runge–Kutta time marching* (RK4C2), *fourth-order centered differences with fourth-order Runge–Kutta time marching* (RK4C4), *and the maximum-order scheme.*

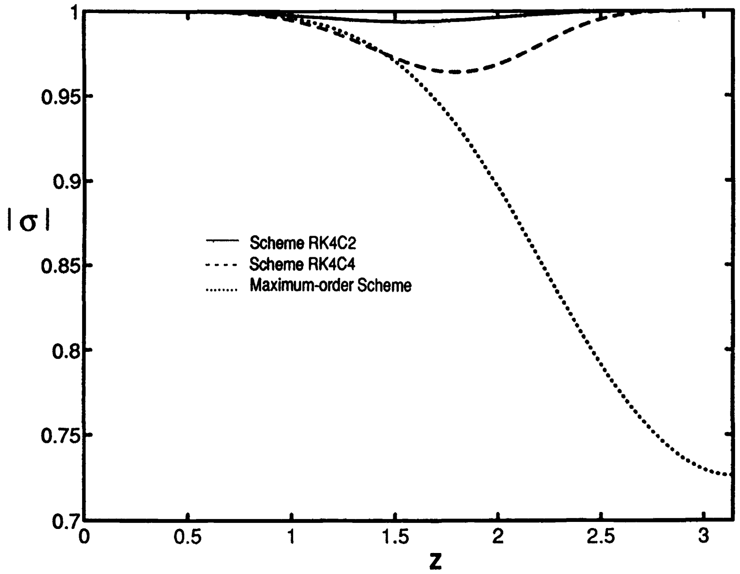


FIG. 2. $|\sigma|$ *for second-order centered differences with fourth-order Runge–Kutta time marching* (RK4C2), *fourth-order centered differences with fourth-order Runge–Kutta time marching* (RK4C4), *and the maximum-order scheme.*

scheme is largest at the "design" error tolerance 2×10^{-5}. For this error tolerance, it produces 1.6 times the resolving efficiency of the maximum-order scheme and four times the resolving efficiency of the fourth-order scheme. For error tolerances below 2×10^{-5}, the optimized

TABLE 1
Resolving efficiency of RK4C2, RK4C4, *and the maximum-order scheme.*

	e =			
Scheme	10^{-5}	2×10^{-5}	10^{-4}	3×10^{-3}
RK4C2	<0.01	<0.01	<0.01	0.04
RK4C4	0.04	0.05	0.07	0.17
Maximum order	0.11	0.12	0.16	0.29

TABLE 2
PPW required for specified error tolerances.

	e =			
Scheme	10^{-5}	2×10^{-5}	10^{-4}	3×10^{-3}
RK4C2	>200	>200	>200	50
RK4C4	50	40	29	12
Maximum order	18	17	13	7

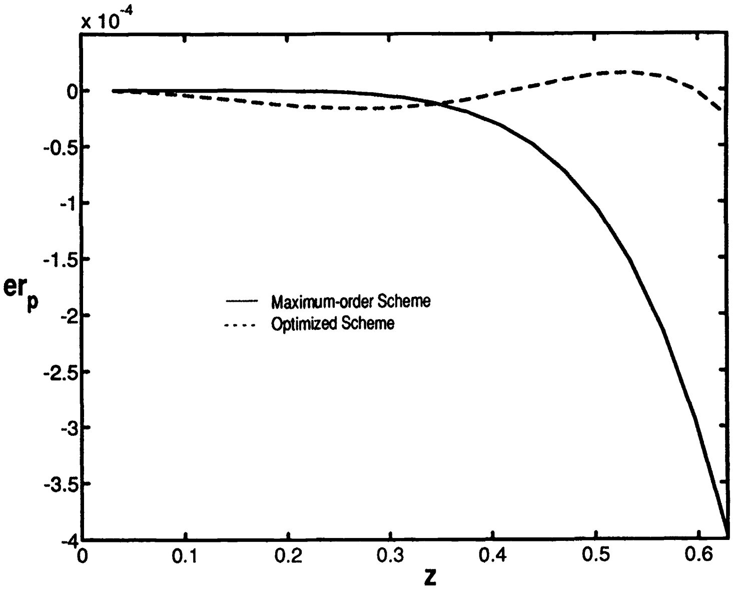


FIG. 3. *Phase error for the maximum-order scheme and the optimized scheme.*

scheme is substantially inferior to the maximum-order scheme. For higher values of error tolerance, the advantage of the optimized scheme is reduced in comparison with both the maximum-order scheme and the fourth-order scheme.

Another useful approach for assessing finite-difference schemes for wave propagation is to determine the *PPW* required to maintain global phase and amplitude errors below a specified level as a function of the number of wavelengths travelled. The global amplitude error is given by

$$Er_a = \left| 1 - |\sigma(z, C)|^{\frac{2\pi n_w}{Cz}} \right|, \tag{22}$$

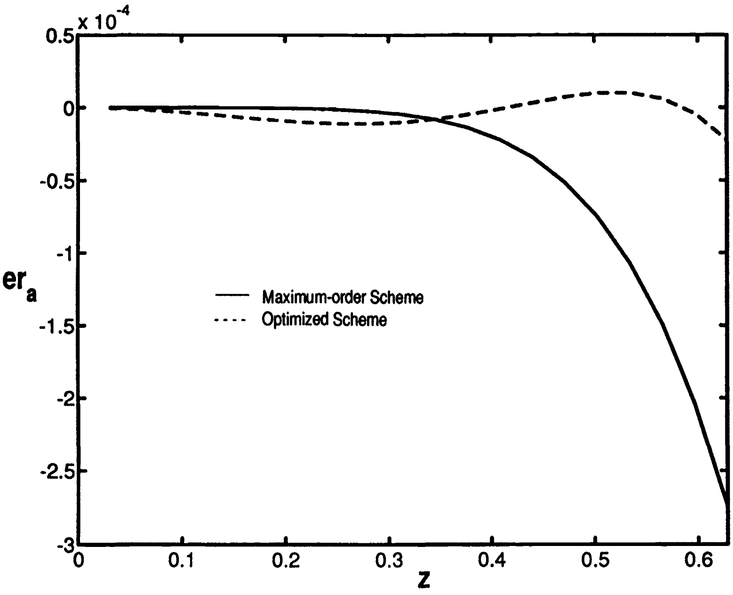


FIG. 4. *Amplitude error for the maximum-order scheme and the optimized scheme.*

TABLE 3
Resolving efficiency and PPW required for optimized scheme.

	e =			
	10^{-5}	2×10^{-5}	10^{-4}	3×10^{-3}
r	0.05	0.20	0.22	0.32
PPW	40	10	9	6

where n_w is the number of wavelengths travelled. The global phase error is

$$Er_p = \frac{2\pi n_w}{Cz}|\phi(C, z) + Cz|. \tag{23}$$

Figure 5 shows the *PPW* required to maintain Er_a and Er_p less than 0.1. The maximum-order scheme can accurately simulate the propagation of waves over distances greater than five hundred wavelengths with a grid resolution of less than twenty points per wavelength. This is less than half of the *PPW* required by the combination of fourth-order centered differences and fourth-order Runge-Kutta time marching. In three dimensions, this translates to more than eight times fewer grid nodes. The optimized scheme is superior for simulations in which waves travel under three hundred wavelengths. For such cases, good accuracy is obtained with roughly ten points per wavelength. In [21] several other finite-difference schemes are compared on this basis.

5. Stability and numerical boundary schemes. From Fourier analysis, the maximum-order fully discrete scheme is stable up to a Courant number of roughly 1.5 in one dimension and the optimized scheme is unconditionally unstable. Stability by Fourier analysis is a necessary condition for Lax-Richtmyer stability. However, the instability of the optimized scheme is very mild and we now show that on a finite domain asymptotic (or time) stability is achieved. Consider a semidiscrete approximation to (8) obtained by dividing the domain into

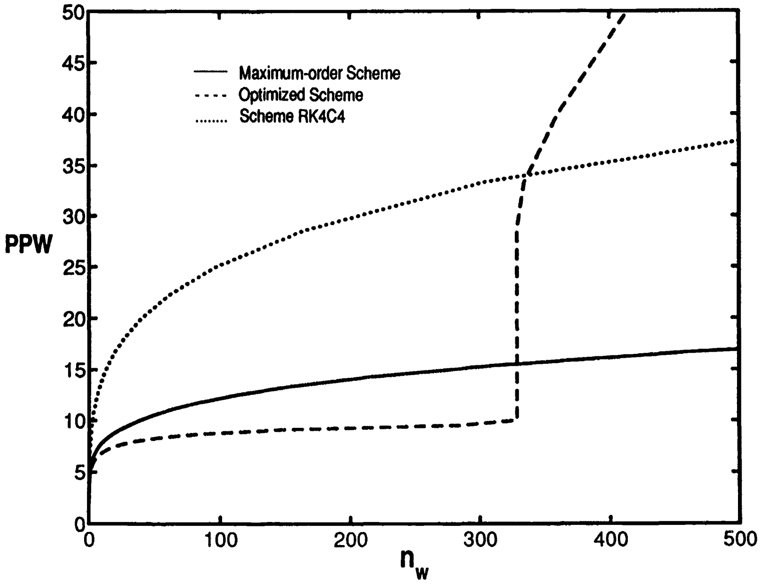


FIG. 5. *PPW requirements for the maximum-order scheme, the optimized scheme, and the combination of fourth-order centered differences and fourth-order Runge–Kutta time marching.*

M subintervals of length $\Delta x = 1/M$ and approximating the spatial derivative in the interior by (1). At the inflow boundary, the derivative is approximated by [18]:

$$(24)\qquad \begin{aligned} (\delta_x u)_1 &= \frac{1}{60\Delta x}[-12u_0 - 65u_1 + 120u_2 - 60u_3 + 20u_4 - 3u_5],\\ (\delta_x u)_2 &= \frac{1}{60\Delta x}[6.6u_0 - 51.6u_1 + 34u_2 - 12u_3 + 39u_4 - 19.6u_5 + 3.6u_6]. \end{aligned}$$

These operators are fifth-order accurate and hence do not compromise the global accuracy of the method. At the outflow boundary, the difference operators for the last three points in the grid are formed using fifth-order space extrapolation together with the interior differencing scheme. The space extrapolation can be written in the form

$$(25)\qquad (1 - E^{-1})^p u_{M+1} = 0,$$

where the shift operator E is defined by $Eu_j = u_{j+1}$ and the order of the approximation is $p - 1$. When this approach is applied to hyperbolic systems, flux-vector splitting is required near boundaries [20].

The semidiscrete form can be written as

$$(26)\qquad \frac{d\mathbf{u}}{dt} = \tilde{\mathbf{A}}\mathbf{u}, \qquad \text{where} \qquad \tilde{\mathbf{A}} = \frac{c}{\Delta x}\mathbf{A},$$

$\mathbf{u} = [u_1, u_2, \ldots, u_{M-1}, u_M]^T$. Figure 6 shows the eigenvalues of $\mathbf{A}$ for the maximum-order and optimized spatial schemes with $M = 100$. Each scheme produces two boundary-condition-dependent eigenvalues [2] but these lie in the left half-plane and hence do not pose a problem. These eigenvalue spectra both lie well within the stability contours of the respective time-marching methods, as shown in Figure 7, and thus the fully discrete methods are asymptotically stable at a Courant number of unity.

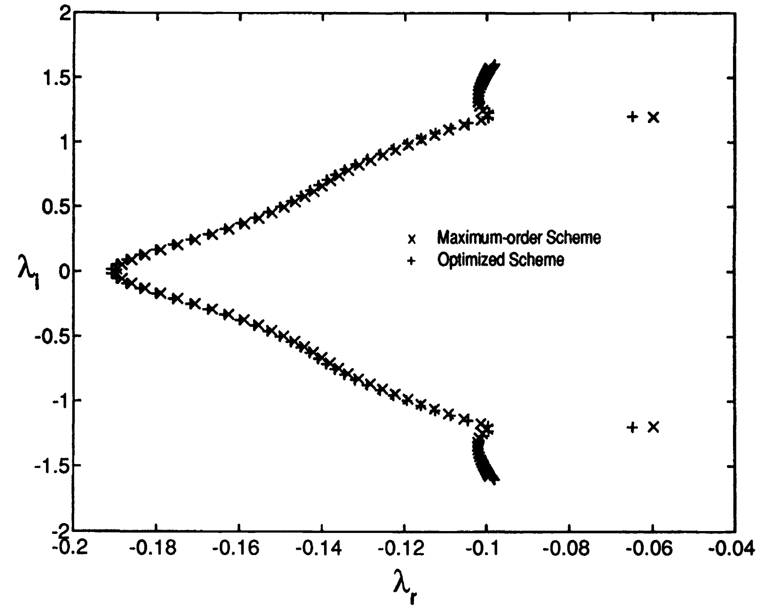


FIG. 6. *Eigenvalue spectra of semidiscrete operators.*

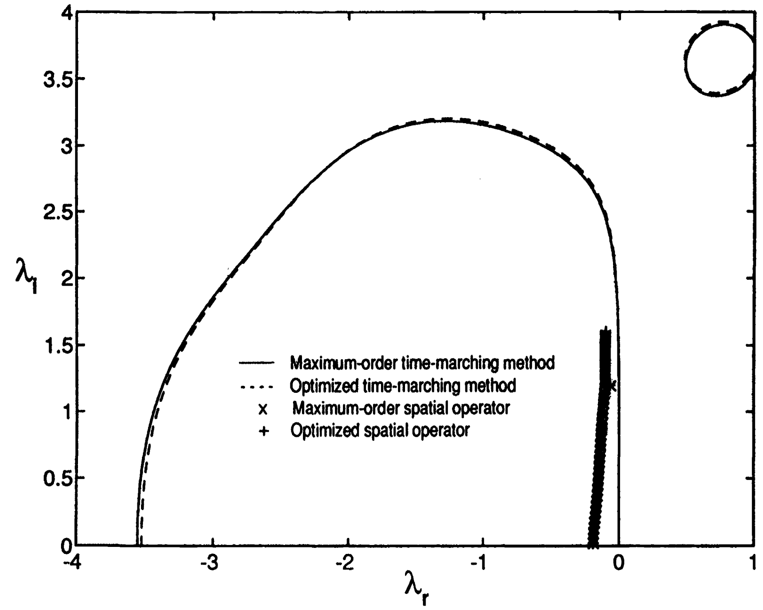


FIG. 7. *Stability contours of the time-marching methods.*

Asymptotic stability is a necessary condition for Lax-Richtmyer stability but it is not sufficient. Thus we now consider the amplification matrix **G** given by

$$\mathbf{G} = \mathbf{I} + C\mathbf{A} + \beta_2(C\mathbf{A})^2 + \beta_3(C\mathbf{A})^3 + \beta_4(C\mathbf{A})^4 + \beta_5(C\mathbf{A})^5 + \beta_6(C\mathbf{A})^6. \quad (27)$$

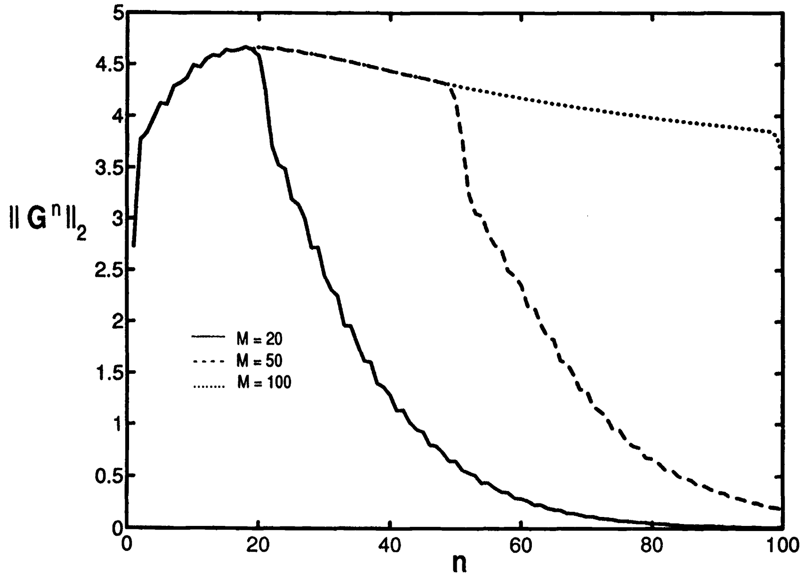


FIG. 8. *Variation of the L_2-norm of the amplification matrix of the maximum-order scheme.*

A necessary and sufficient condition for stability of a fully discrete finite-difference scheme is that there exists a constant $K \geq 1$ such that

$$\|(\mathbf{G}(\Delta x, h))^n\| \leq K \tag{28}$$

for all $n \geq 0$, $0 \leq nh \leq T$ with T fixed. For hyperbolic problems, the Courant number must be kept constant as n is increased. Figure 8 shows $\|\mathbf{G}^n\|_2$ for the maximum-order scheme for three different values of M and a Courant number of unity. Figure 9 shows similar results for the optimized scheme. In both cases, $\|\mathbf{G}^n\|_2$ is clearly bounded and hence both schemes appear to be stable. This is consistent with the results of [20], in which both schemes were used for simulations of electromagnetic waves with no evidence of instability. However, a singular value decomposition of $\mathbf{G}^n$ shows that the growth in $\|\mathbf{G}^n\|$ shown in Figure 9 is associated with the numerical boundary scheme at the inflow boundary. This obscures the very slow growth of the unstable modes of the optimized scheme, which is revealed if the numerical boundary scheme at inflow is removed. Nevertheless, Figure 9 provides some reassurance that the instability causes no immediate difficulties. As shown in Figure 4, the maximum value of $|\sigma|$ predicted using Fourier analysis is 1.00002 at a Courant number of unity. Since $1.00002^{34,000} < 2$, the scheme can safely be used for well over 34,000 time steps without the solution exhibiting any instability. However, the optimized scheme is not intended for such long simulations since the maximum-order scheme becomes more accurate if the distance travelled becomes very large, as shown in Figure 5.

6. Numerical experiments. We now consider several numerical experiments in order to further compare the four schemes discussed above. The one-dimensional linear convection equation with $c = 1$ is solved with periodic boundary conditions on the domain $0 \leq x \leq 1$. The initial condition is given by a Gaussian-modulated cosine function as follows:

$$U(x, 0) = (\cos \kappa x)e^{-0.5[(x-0.5)/\sigma_g]^2}$$

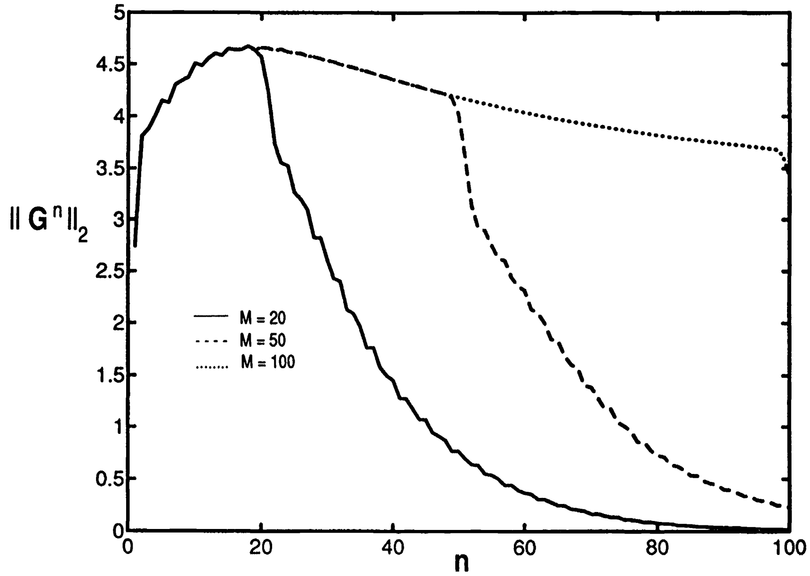


FIG. 9. *Variation of the L_2-norm of the amplification matrix of the optimized scheme.*

with various values of σ_g and κ. For the present schemes, the errors in the practical range of wavenumbers are generally largest at $\theta = 0^o$ and decrease with decreasing Courant number. Therefore, one-dimensional numerical experiments with a Courant number of unity represent the worst case.

For the first experiment, we consider a Gaussian ($\kappa = 0$) with $\sigma_g = 0.04$. The grid is uniform with 100 points across the domain. Figure 10 shows the result after 100 time steps with a Courant number of unity, i.e., at $t = 1$. The exact solution is identical to the initial condition. The solution produced by scheme RK4C2 is poor, while the other schemes are very accurate. After 1000 time steps, the solution produced by RK4C4 is inadequate for many purposes, while the maximum-order and optimized schemes remain very accurate, as shown in Figure 11.

Figures 12–15 compare the maximum-order and optimized schemes for Gaussian initial conditions with four different values of σ_g. The results are again shown for $C = 1$ and $t = 1$. The figures show the difference between the computed solution and the exact solution. For $\sigma_g = 0.04$ and $\sigma_g = 0.03$, the L_2-norm of the error produced by the optimized scheme is less than half of that produced by the maximum-order scheme. For $\sigma_g = 0.02$ the improvement from the optimized scheme is reduced and for $\sigma_g = 0.05$ the optimized scheme produces more error than the maximum-order scheme.

These results can be better understood by considering the normalized power spectra of these Gaussians, which are shown in Figure 16. These spectra show the wavenumber content of the Gaussians as a function of z based on a 100-point grid. They can be compared with Figures 3 and 4, which show the numerical errors also as a function of z. For example, Figures 3 and 4 show that the optimized scheme is much more accurate than the maximum-order scheme for roughly $0.4 \leq z \leq 0.7$. With $\sigma_g = 0.05$, there is virtually no content in this region and hence the optimized scheme is inferior to the maximum-order scheme since it produces more error at low wavenumbers. For $\sigma_g = 0.04$ and $\sigma_g = 0.03$, there is some content in the range $0.4 \leq z \leq 0.7$ and little content at higher wavenumbers. Consequently, the error is dominated by these modes and the optimized scheme is superior. Finally, for $\sigma_g = 0.02$, the

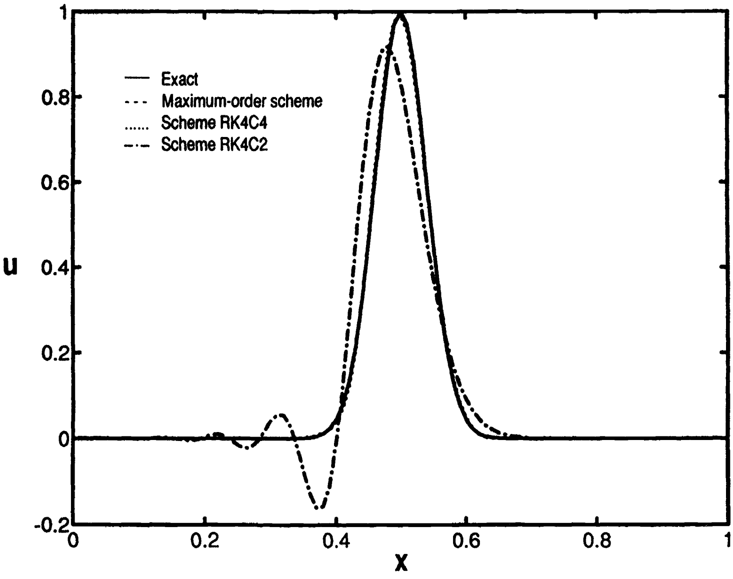


FIG. 10. *Solution at* $t = 1$.

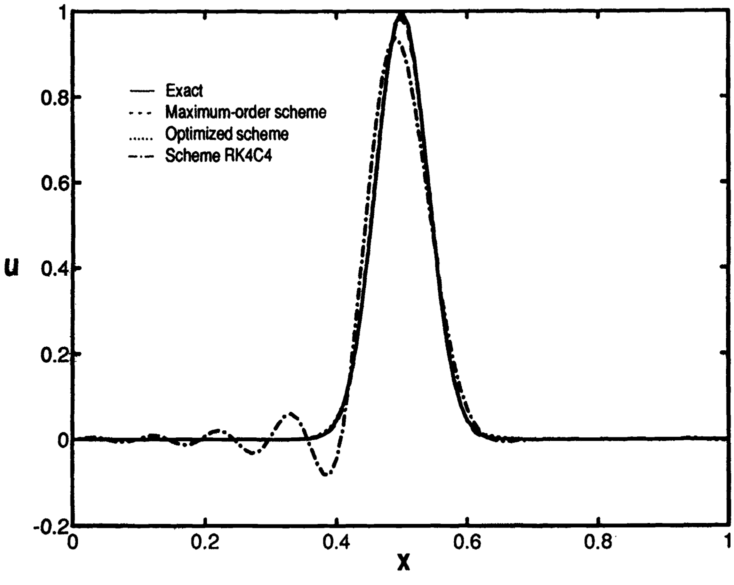


FIG. 11. *Solution at* $t = 10$.

error is dominated by wavenumbers with $z > 0.7$ and thus the optimized scheme is not much superior to the maximum-order scheme.

The gains produced by the optimized scheme for Gaussians are quite modest, even for $\sigma_g = 0.03$ and $\sigma_g = 0.04$. This occurs because Gaussians always have considerable low wavenumber content, which is convected more accurately by the maximum-order scheme. The optimized scheme is more effective for functions with a narrower bandwidth. Figures 17–19 show results for Gaussian-modulated cosine functions with $\sigma_g = 0.1$ and $\kappa = 24\pi$,

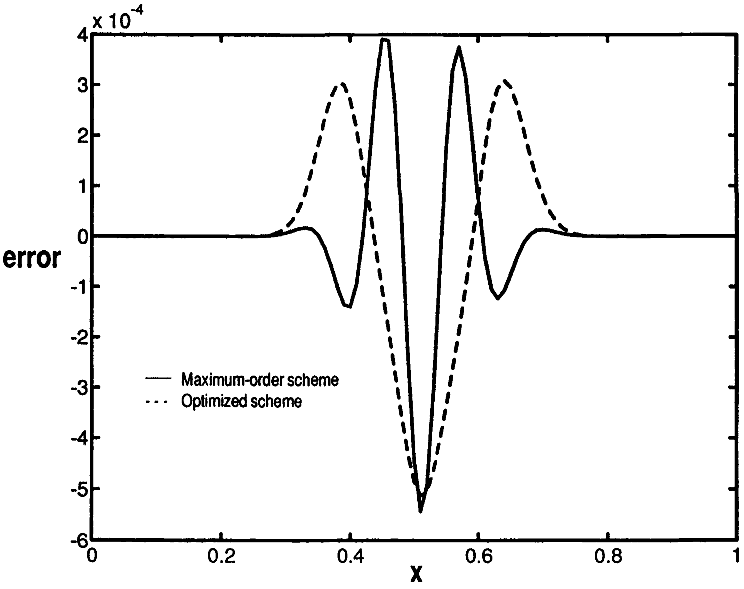


FIG. 12. *Error for* $\sigma_g = 0.05$.

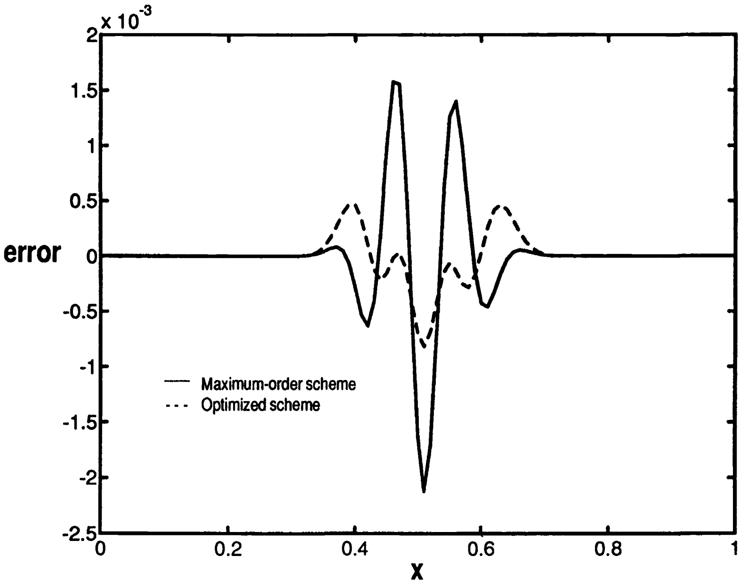


FIG. 13. *Error for* $\sigma_g = 0.04$.

32π, and 40π, on a 200-point grid. The results are shown after 200 time steps at a Courant number of unity, i.e., at $t = 1$. The corresponding normalized power spectra are shown in Figure 20. In each case, these functions have considerable spectral content in the range for which the optimized scheme is superior. For $\kappa = 32\pi$, the L_2-norm of the error produced by the optimized scheme is more than ten times less than that produced by the maximum-order scheme.

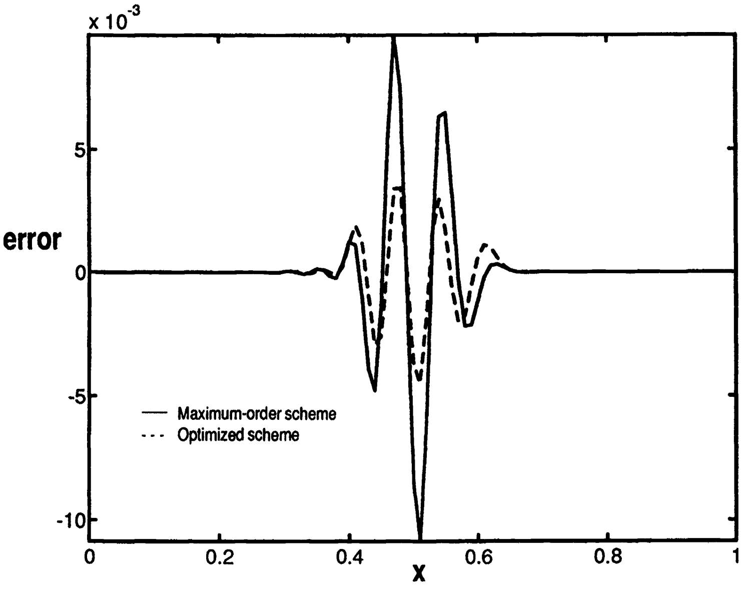


FIG. 14. *Error for* $\sigma_g = 0.03$.

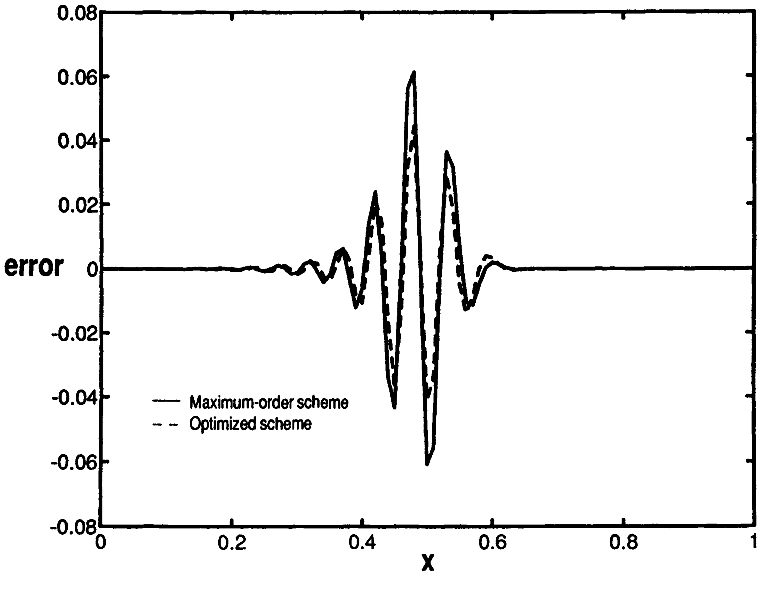


FIG. 15. *Error for* $\sigma_g = 0.02$.

7. Discussion. In this section, we discuss some considerations in selecting and developing finite-difference schemes for wave propagation problems. Initially an error tolerance must be determined. This is based on two factors: (1) the level of accuracy required of the simulation in order to produce meaningful results and (2) an estimate of the largest distance a wave will travel during the simulation. Next an estimate must be made of the shortest wavelength which must be accurately resolved. This is also based on the accuracy requirements of the sim-

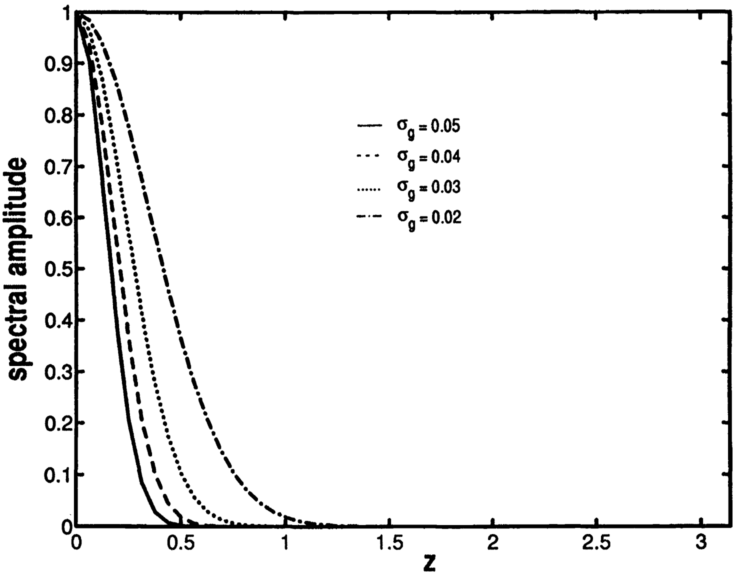


FIG. 16. *Normalized power spectra of Gaussians.*

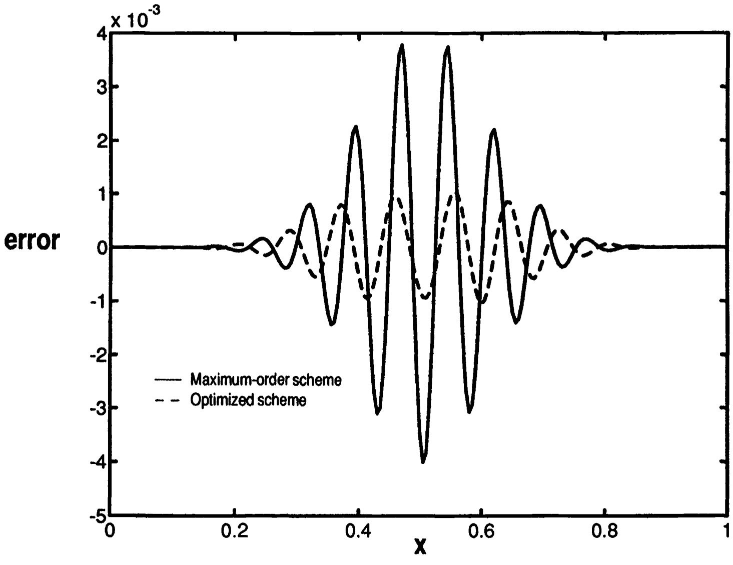


FIG. 17. *Error for* $\kappa = 24\pi$.

ulation. For a given scheme, the grid spacing can then be determined to produce the required accuracy in phase and amplitude for the shortest wavelength of interest based on Fourier error analysis. The grid resolution must be sufficient to satisfy the accuracy requirements for the worst combination of Courant number and propagation direction.

The compromise involved in optimizing a scheme is clearly revealed in Figure 5. For small distances of travel, the optimized scheme is superior but as the distance is increased the maximum-order scheme eventually requires fewer *PPW*. [21] includes a scheme similar to

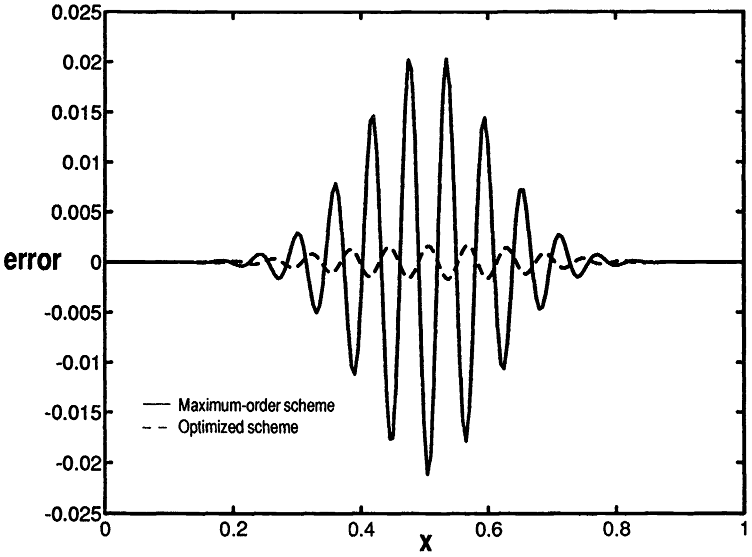


FIG. 18. *Error for* $\kappa = 32\pi$.

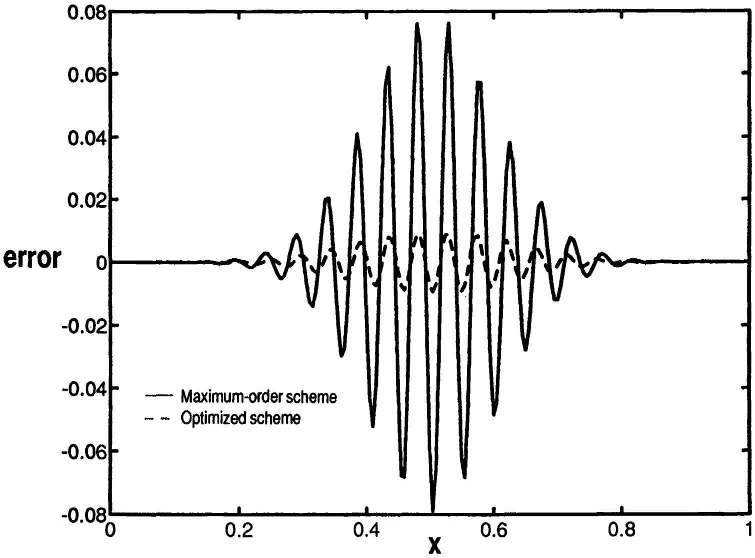


FIG. 19. *Error for* $\kappa = 40\pi$.

that presented here which is optimized for waves resolved with at least 7.5 *PPW*. This scheme is slightly superior to the present optimized scheme for distances of travel less than roughly 75 wavelengths but is inferior for longer distances.

For some wave propagation applications, it is essential that the numerical scheme produce no dissipation, i.e., no amplitude error. The schemes presented here are inappropriate for such problems. However, for most problems it is sufficient that the amplitude error be less than or comparable to the phase error. Furthermore, the damping of high wavenumber modes produced by the present schemes can be helpful in some applications [17]. If reduced dissipation is

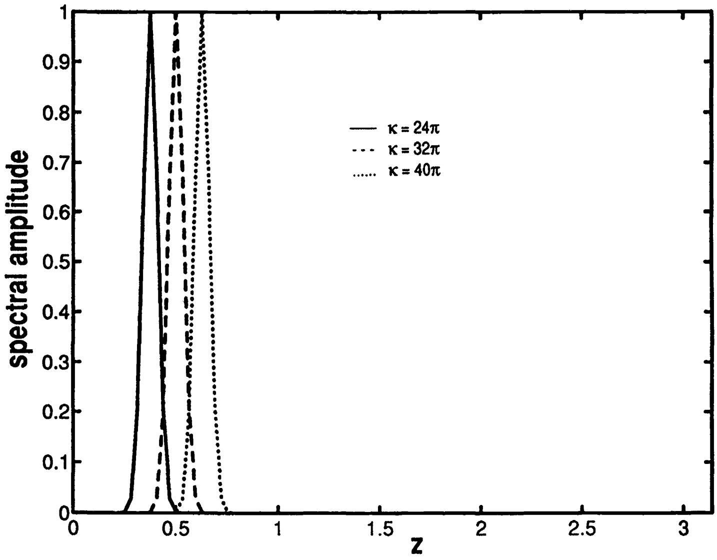


FIG. 20. *Normalized power spectra of Gaussian-modulated cosine functions.*

desired, the present spatial scheme can be used without the filter. Fourth-order Runge-Kutta time marching should then be used for stability.

8. Conclusions. The two finite-difference schemes presented here provide a promising option for simulating long-range propagation of linear waves. Potential applications include electromagnetics and acoustics. Both schemes combine a seven-point spatial operator and an explicit six-stage low-storage time-marching method of the Runge-Kutta type. The optimized scheme was developed by minimizing the maximum phase and amplitude errors for waves which are resolved with at least ten points per wavelength. The maximum-order scheme can accurately simulate the propagation of waves over distances greater than five hundred wavelengths with a grid resolution of less than twenty points per wavelength. The optimized scheme is intended for simulations in which waves travel under three hundred wavelengths. For such cases, good accuracy is obtained with roughly ten points per wavelength. Future work will address the application of these schemes to complex geometries.

REFERENCES

[1] R. M. ALFORD, K. R. KELLY, AND D. M. BOORE, *Accuracy of finite-difference modeling of the acoustic wave equation*, Geophysics, 39 (1974), pp. 834-842.

[2] R. M. BEAM AND R. F. WARMING, *The asymptotic spectra of banded Toeplitz and quasi-Toeplitz matrices*, SIAM J. Sci. Comput., 14 (1993), pp. 971-1006.

[3] C. L. CHEN, S. R. CHAKRAVARTHY, AND B. L. BIHARI, *Numerical Solution of Acoustic Equations with Unstructured Grids Using a CFD-Based Approach*, AIAA Paper 92-2698, American Institute of Aeronautics and Astronautics, New York, 1992.

[4] G. COHEN AND P. JOLY, *Fourth order schemes for the heterogeneous acoustics equation*, Comput. Methods Appl. Mech. Engrg., 80 (1990), pp. 397-407.

[5] M. A. DABLAIN, *The application of high-order differencing to the scalar wave equation*, Geophysics, 51 (1986), pp. 54-66.

[6] S. DAVIS, *Matrix-Based Finite Difference Algorithms for Computational Acoustics*, AIAA Paper 90-3942, American Institute of Aeronautics and Astronautics, New York, 1990.

[7] ———, *Low-dispersion finite difference methods for acoustic waves in a pipe*, J. Acoust. Soc. Amer., 90 (1991), pp. 2775-2781.

[8] O. Holberg, *Computational aspects of the choice of operator and sampling interval for numerical differentiation in large-scale simulation of wave phenomena*, Geophys. Prospecting, 35 (1987), pp. 629-655.

[9] S. K. Lele, *Compact finite difference schemes with spectral-like resolution*, J. Comput. Phys., 103 (1992), pp. 16-42.

[10] K. J. Marfurt, *Accuracy of finite-difference and finite-element modeling of the scalar and elastic wave equations*, Geophysics, 49 (1984), pp. 533-549.

[11] P. Sguazzero, M. Kindelan, and A. Kamel, *Dispersion-bounded numerical integration of the elastodynamic equations with cost-effective staggered schemes*, Comput. Methods Appl. Mech. Engrg., 80 (1990), pp. 165-172.

[12] V. Shankar, A. H. Mohammadian, and W. F. Hall, *A time-domain finite-volume treatment for the Maxwell's equations*, Electromagnetics, 10 (1990), pp. 127-147.

[13] A. Taflove, *Re-inventing Electromagnetics: Supercomputing Solution of Maxwell's Equations Via Direct Time Integration on Space Grids*, AIAA Paper 92-0333, American Institute of Aeronautics and Astronautics, New York, 1992.

[14] C. K. W. Tam, *Discretization Errors Inherent in Finite Difference Solution of Propellor Noise Problems*, AIAA J., 30 (1992), pp. 608-615.

[15] C. K. W. Tam and J. C. Webb, *Dispersion-relation-preserving finite difference schemes for computational acoustics*, J. Comput. Phys., 107 (1993), pp. 262-281.

[16] R. Vichnevetsky and J. B. Bowles, *Fourier Analysis of Numerical Approximations of Hyperbolic Equations*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1982.

[17] M. Yarrow, J. A. Vastano, and H. Lomax, *Pulsed Plane Wave Analytic Solutions for Generic Shapes and the Validation of Maxwell's Equations Solvers*, AIAA Paper 92-0016, American Institute of Aeronautics and Astronautics, New York, 1992.

[18] D. W. Zingg and H. Lomax, *On the eigensystems associated with numerical boundary schemes for hyperbolic equations*, in Numerical Methods for Fluid Dynamics, M. J. Baines and K. W. Morton, eds., Clarendon Press, Oxford, 1993, pp. 473-481.

[19] ———, *Some Aspects of High-Order Numerical Solutions of the Linear Convection Equation with Forced Boundary Conditions*, AIAA Paper 93-3381 in the Proc. of the 11th AIAA Computational Fluid Dynamics Conf., American Institute of Aeronautics and Astronautics, New York, 1993.

[20] D. W. Zingg, P. D. Giansante, and H. M. Jurgens, *Experiments with High-Accuracy Finite-Difference Schemes for the Time-Domain Maxwell Equations*, AIAA Paper 94-0232, American Institute of Aeronautics and Astronautics, New York, 1994.

[21] D. W. Zingg, E. M. Epstein, and H. M. Jurgens, *A Comparison of Finite-Difference Schemes for Computational Aeroacoustics*, AIAA Paper 95-0162, American Institute of Aeronautics and Astronautics, New York, 1995.

SIAM J. SCI. COMPUT.
Vol. 17, No. 2, pp. 347-357, March 1996

A NONLINEAR, SUBGRIDSCALE MODEL FOR INCOMPRESSIBLE VISCOUS FLOW PROBLEMS*

WILLIAM J. LAYTON†

Abstract. We consider a nonlinear subgridscale model of the Navier-Stokes equations resulting in a Ladyzhenskaya-type system. The difference is that the power "p" and scaling coefficient $\mu(h) \doteq O(h^\sigma)$ do not arise from macroscopic fluid properties and can be picked to ensure both L^∞-stability and yet be of the order of the basic discretization error in smooth regions. With a properly scaled p-Laplacian-type artificial viscosity one can construct a higher-order method which is just as stable as first-order upwind methods.

Key words. high Reynolds number, subgridscale model, Ladyzhenskaya model

AMS subject classifications. Primary 76M10; Secondary 65N30

1. Introduction. Consider the approximate solution of steady, incompressible, viscous flow problems, as described by the Navier-Stokes equations for the velocity $\mathbf{u} : \Omega \subset \mathbb{R}^d \to \mathbb{R}^d (d = 2, 3)$ and the pressure $\lambda : \Omega \to \mathbb{R}$:

$$(1.1) \qquad -\operatorname{Re}^{-1}\Delta\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u} + \nabla\lambda = \mathbf{f} \text{ in } \Omega, \quad \nabla\cdot\mathbf{u} = 0 \text{ in } \Omega,$$

subject to the usual no-slip boundary condition and normalization condition for λ:

$$(1.2) \qquad \mathbf{u} = 0 \text{ on } \partial\Omega, \ \int_\Omega \lambda dx = 0.$$

In (1.1), (1.2) Ω is a bounded, polygonal domain in $\mathbb{R}^d$, Re is the Reynolds number, λ is the fluid pressure, and $\mathbf{u}$ is the fluid velocity. To fix ideas we consider a (centered) finite element discretization of (1.1), (1.2) with a global (or outer) meshwidth (meaning maximum d-simplex diameter) "h."

We also specifically consider herein the case of "high" Reynolds numbers. For Re high enough, the equilibrium problem (1.1), (1.2) loses its stability. Thus, in this report large Re will be understood to mean *large with respect the number of degrees of freedom available for the approximation of* (1.1), (1.2). Thus $0 < \operatorname{Re}^{-1} \leq O(h)$ will be the case considered.

In fluid flow problems, considerable energy in the flow can be contained in "eddies, vortices/solution scales," etc. which are not large enough to be represented on a computationally feasible grid. It is common practice for such problems to employ either a specialized discretization such as [16, 19, 28, 29] or some sort of "subgridscale" (SGS) model to represent these effects on the grid, at least in some averaged sense. The goal is to remove the "excess energy" from the large (resolved) eddies and to return some energy to the large eddies when the eddy moves into a region with smaller local meshwidth. One common approach (others are certainly possible; see Dubois, Jauberteau, and Temam [10] and Heywood and Rannacher [17]) is to obtain closure in the resulting set of SGS averaged equations via a judicious combination of continuum mechanical principles and phenomenology. Often this approach results in the same effect as beginning with, for example, an algebraic turbulence model, such as studied in Gunzburger and Turner [14], and setting the characteristic length scale $\ell(x)$ to be the local meshwidth $h(x)$; for this approach see, e.g., Leslie and Quarini [23]. Thus, SGS modelling, while unavoidable in many applications, can be highly arguable.

*Received by the editors January 31, 1994; accepted for publication (in revised form) October 5, 1994. This work was partially supported by NSF grant 9400057.

†Institute for Computational Mathematics and Applications, Department of Mathematics, University of Pittsburgh, Pittsburgh, PA 15260 (WJL@VMS.CIS.PITT.EDU).

This paper takes a much more modest approach, not linked to either phenomenology or (alas) mechanics, but having similar motivation and goals. To motivate the present proposal consider first the usual $O(h)$ upwind/artificial viscosity approximation to (1.1), (1.2). This may be considered to be the most successful, although crudest, SGS model in practical calculations. Effects smaller than one mesh cell are spread to $O(\sqrt{h})$ distance thus resolvable on the outer mesh. It "stabilizes" both the discretization and the resulting linear system and does not alter the graph norm error (H^1 for velocities, L^2 for pressures) when using linear elements. Although (1.1), (1.2) does not satisfy a maximum principle,[1] the upwind/artificial viscosity discretization does control "wiggles" since L^∞-stability of the upwind approximation is easily proven using discrete Sobolev inequalities:

$$\sup_{0<\mathrm{Re}<\infty} \|\mathbf{u}^h\|_{L^\infty} \leq C(\Omega, f)h^{-\frac{d}{2}}|lnh|^{1-\frac{1}{d}}, \ d = \dim(\Omega) = 2, 3. \tag{1.3}$$

(The Navier-Stokes equations do possess singular solutions which are not in $L^\infty_{\mathrm{loc}}(\Omega)$ on small sets, Boisvert and Ames [4], so the constant on the right-hand side of (1.3) must $\to \infty$ as $h \to 0$ if the method is convergent.) Further, other mathematical support for some advantages of using upwind/artificial viscosity discretizations can be given [12, 13, 28, 29].

From the point of view of this paper, an SGS model should share these good attributes while at least partially overcoming the well-documented accuracy and resolution shortcomings of upwinding and first-order artificial viscosity. Therefore, relevant criteria include the following.

(i) The approximate solution should satisfy the same L^∞-stability bound (1.3) as the upwind method.

(ii) For degree k velocity elements and in smooth flow regions, the model should involve an $O(h^k)$ or $O(h^{k+1})$ perturbation from a consistent variational formulation.

(iii) Mathematical support for the agreeable effects of the model should be possible (at least for "model" problems).

(iv) The additional terms arising from the SGS model should be "stabilizing" for the algebraic system in the sense of adding contractive terms. This plays the role of removing "excess energy" from the large eddies.

To obtain a bound of the form (1.3) via the Sobolev embedding theorem, only three parameters are available: d the dimension of Ω, m the order of the derivative, and p the L^p-norm in which smoothness is measured. d is fixed at $\dim(\Omega)$ (2 or 3) and m may be considered fixed at 2 (not wishing to increase the order of (1.1) for the usual reasons). Thus, only p is available, leading to a p-Laplacian! An additional, nonlinear, artificial viscosity term of the form

$$AV_p(\mathbf{u}^h, \mathbf{v}) := \mu(h)(|\nabla\mathbf{u}^h|^{p-2}\nabla\mathbf{u}^h, \nabla\mathbf{v})$$

is thus suggested, where $(\cdot, \cdot)$ and $\|\cdot\|$ denote the usual $(L^2(\Omega))^d$ inner product and norm. For μ fixed (e.g., $\mu(h) \leq 0(h^\sigma)$, $\sigma =$ polynomial degree $(X^h) + 1$), taking $2 \leq p < \infty$ large enough will satisfy the conditions (i)-(iv) above.

The above form of "h" is taken to simplify the analysis somewhat. If meshwidths are highly variable then $AV_p(\cdot, \cdot)$ should be modified appropriately to use the correct local meshwidth:

$$AV_p(\mathbf{u}^h, \mathbf{v}) = \sum_{\substack{\text{all triangles} \\ e}} \int_e \mu \text{ (diameter } (e))|\nabla\mathbf{u}^h|^{p-2}\nabla\mathbf{u}^h : \nabla\mathbf{v}dx,$$

[1] Similarity solutions are known with point singularities [4]. The singular set in space-time of solutions to the Navier-Stokes equation has $1 - D$ Hausdorff dimension zero [6], so it is thin.

with $\mu(h)$ given by the following formula. Choose $p > \frac{d}{2}$ and $(k+1) \leq \sigma \leq p-1$ if $d=2$ and $(k+1) \leq \sigma \leq \frac{3}{2}(p-1)$ if $d=3$. Pick (μ_0 = constant and h = local meshwidth)

$$\mu(h) = \mu_0 |lnh|^{-(\frac{d-1}{d})(p-1)} h^{\sigma}. \tag{1.4}$$

This choice is motivated in Lemma 2.2, §2, but optimality of (1.4) can only be determined by more testing. Note that we may choose σ as large as we want (for example σ = polynomial degree of finite element space +1) by taking p correspondingly large: $p > \frac{2}{3}\sigma + 1$ in three dimensions.

The need for agreeable SGS models is even more acutely felt when the (nonlinear or linearized) discrete Navier-Stokes equations are solved using a multilevel method. For example, the finest mesh may indeed be sufficient to resolve solution scales. However, those same scales can possibly be troublesome on coarser meshes. This can be potentially troublesome for a two-level or multilevel discretization of nonlinear problems when the coarse mesh approximation should be in the ball of attraction for Newton's method for the fine mesh problem. (These promising methods were used already in 1935 by Southwell [27]. Their mathematical validation has recently been pioneered by Xu [31] for elliptic boundary value problems, and then generalized to the Navier-Stokes equations in [19-21].) The "Navier-Stokes plus p-Laplacian" equations were originally proposed by Ladyzhenskaya [18] as an alternate model for fluids with large stresses. In our view, the Ladyzhenskaya alteration is *quite* promising as an SGS model with (1.4) when not justified as a macroscopic fluid model. Finite element error analysis of this model was carried out in Du and Gunzburger [7-9] under a global uniqueness (small data) condition. The preliminary analysis given herein thus complements that of [7-9] in that we give error analysis in §3 for high Re with respect to solutions of (1.1) with $\mu = \mu(h) \to 0$ as $h \to 0$.

2. Preliminaries and max-norm stability. We consider the nonlinear SGS method: seek $(\mathbf{u}^h, \lambda^h) \in (X^h, Q^h)$ satisfying

$$AV_p(\mathbf{u}^h, \mathbf{v}) + \mathrm{Re}^{-1}(\nabla \mathbf{u}^h, \nabla \mathbf{v}) + b(\mathbf{u}^h, \mathbf{u}^h, \mathbf{v}) - (\lambda^h, \nabla \cdot \mathbf{v}) + (q, \nabla \cdot \mathbf{u}^h) = (f, \mathbf{v}) \tag{2.1}$$

for all $(\mathbf{v}, q) \in (\mathbf{x}^h, \mathbf{Q}^h)$, where $AV_p(\cdot,\cdot)$ is given, as indicated in the introduction, by

$$AV_p(\mathbf{u}, \mathbf{v}) := \mu(h)(|\nabla \mathbf{u}|^{p-2} \nabla \mathbf{u}, \nabla \mathbf{v}).$$

Here $p = 2$ corresponds to usual linear artificial viscosity, and $b(\mathbf{u}, \mathbf{v}, \mathbf{w})$ is the usual skew-symmetrized trilinear form:

$$b(\mathbf{u}, \mathbf{v}, \mathbf{w}) := \frac{1}{2} \int_{\Omega} [\mathbf{u} \cdot \nabla \mathbf{v} \cdot \mathbf{w} - \mathbf{u} \cdot \nabla \mathbf{w} \cdot \mathbf{v}] dx.$$

We shall assume that $\mathbf{X}^h$ satisfies the usual inverse and discrete Sobolev inequalities. In particular, for $d = 2, 3$ and all $\mathbf{v} \in \mathbf{X}^h$ (where $\|\cdot\|$ is the L^2-norm),

$$\|\mathbf{v}\|_{L^\infty} \leq C|ln(h)|^{1-\frac{1}{d}} h^{-(\frac{d-2}{2})} \|\nabla \mathbf{v}\|. \tag{2.2}$$

Additionally, we shall need the $L^p - L^2$-type inverse inequality which follows.

LEMMA 2.1. *Let θ_0 be the minimum angle in the triangulation and $M^k = \{v(x) : v(x)|_e \in \mathcal{P}_k(e)$ for all $e \in \Pi^h(\Omega)\}$, $\mathcal{P}_k$ being the polynomials of degree $\leq k$. Then, for ∇^h the element-wise defined gradient operator, there is a $C = C(\theta_0, p, k)$ such that for $2 \leq p < \infty$, $d = 2, 3$ and all $v \in M^k$,*

$$\|\nabla^h v\|_{L^p(\Omega)} \leq C h^{\frac{d}{2}(\frac{2-p}{p})} \|\nabla^h v\|. \tag{2.3}$$

Proof. First a local version of (2.3) is proven by a scaling argument, using the technique of Dupont and Scott [11]. Let $\hat{e}$ denote the reference element. First consider the affine map $\mathbf{x} = h_e\tilde{\mathbf{x}} + \mathbf{b}$ mapping e to an element $\tilde{e}$ the size of the reference element $\hat{e}$.

A change of variables gives immediately that $\|\nabla v\|_{L^2(e)} = h_e^{(d-2)/2}\|\tilde{\nabla} v\|_{L^2(\tilde{e})}$ (well known) and $\|\tilde{\nabla} v\|_{L^p(\tilde{e})} = h_e^{(p-d)/p}\|\nabla v\|_{L^p(e)}$ (completely analogous). The effect of the shape distortion between $\tilde{e}$ and $\hat{e}$ is accounted for by a compactness argument (following [11]) which contributes a constant $C = C(\theta_0, k)$. Finally, norm equivalence implies $\|\hat{\nabla} v\|_{L^p(\hat{e})} \leq C(p,k,d)\|\hat{\nabla} v\|_{L^2(\hat{e})}$. These combine to yield

$$\|\nabla v\|_{L^p(e)} \leq Ch^{\frac{d}{2}(\frac{2-p}{p})}\|\nabla v\|_{L^2(e)}.$$

The global bound is obtained by the Cauchy-Schwarz inequality as follows. Let $p > 2$ be an integer. Then,

$$\begin{aligned}
\|\nabla^h v\|_{L^p} :&= \left[\sum_e \|\nabla v\|^p_{L^p(e)}\right]^{\frac{1}{p}} \leq Ch^{\frac{d}{2}(\frac{2-p}{p})}\left[\sum_e \|\nabla v\|^p_{L^2(e)}\right]^{1/p} \\
&= Ch^{\frac{d}{2}(\frac{2-p}{p})}\left[\sum_e \|\nabla v\|_{L^2(e)}\|\nabla v\|^{p-1}_{L^2(e)}\right]^{\frac{1}{2}} \leq (C.B.S.\text{ ineq.}) \\
&\leq Ch^{\frac{d}{2}(\frac{2-p}{p})}\|\nabla v\|^{\frac{1}{p}}_{L^2(\Omega)}\left[\sum_e \|\nabla v\|^{p-1}_{L^2(e)}\right]^{\frac{1}{p}} \\
&\text{repeat } p-3 \text{ more times } \leq Ch^{\frac{d}{2}(\frac{2-p}{p})}\|\nabla^h v\|_{L^2(\Omega)}.
\end{aligned}$$

This proves the result for the $2 \leq p < \infty$ integer. The case of noninteger p follows by interpolation. □

LEMMA 2.2 (max-norm stability). *Let $S^h(\Omega)$ satisfy the inverse estimates* (2.2) *and* (2.3). *Then the following holds.*

(i) Linear artificial viscosity. *If $p = 2$ and $\mu(h) = \mu_0 h$, then u^h satisfies $(\mu_0 h + \mathrm{Re}^{-1})\|\nabla u^h\|_0 \leq C\|f\|_{-1}$ so that*

$$\|\mathbf{u}^h\|_{L^\infty} \leq C(k,\Omega)\left(\mu_0 + \frac{\mathrm{Re}^{-1}}{h}\right)^{-1} h^{\frac{-d}{2}}|lnh|^{1-\frac{1}{d}}\ \|\mathbf{f}\|_{-1}.$$

(ii) Nonlinear artificial viscosity. *Let* $\mathrm{Re}^{-1} \leq Ch$. *If $p > \frac{d}{2}$, $\mu = \hat{\mu}h^\sigma$ with $\sigma \leq \frac{d}{2}(p-1)$ (i.e., $p \geq \frac{2\sigma}{d} + 1$), then*

$$\|\mathbf{u}^h\|_{L^\infty} \leq C(k,p,\Omega)\cdot\mu_0^{-\frac{1}{p-1}}h^{-\frac{d}{2}}\|\mathbf{f}\|^{\frac{1}{p-1}}_{W^{q,-1}}.$$

Remark. This suggests choosing $\hat{\mu} = 0(|lnh|^{-((d-1)/d)(p-1)})$, which gives the formula (1.4) of the introduction.

Proof. Set $\mathbf{v} = \mathbf{u}^h$ and $q = \lambda^h$ in (2.1):

$$\mu h^\sigma \int_\Omega |\nabla\mathbf{u}^h|^{p-2}|\nabla\mathbf{u}^h|^2 dx + \mathrm{Re}^{-1}\|\nabla\mathbf{u}^h\|^2_{L^2} = (\mathbf{f},\mathbf{u}^h) \leq \|\mathbf{f}\|_{W^{q,-1}}\|\nabla\mathbf{u}^h\|_{L^p},$$

where $\frac{1}{p} + \frac{1}{q} = 1$. Thus

$$\mu_0 h^\sigma\|\nabla\mathbf{u}^h\|^{p-1}_{L^p} \leq \|\mathbf{f}\|_{W^{q,-1}} \quad \text{and} \quad \|\nabla\mathbf{u}^h\|_{L^p} \leq \mu_0^{-\frac{1}{p-1}}h^{-\frac{\sigma}{p-1}}\|\mathbf{f}\|^{\frac{1}{p-1}}_{W^{q,-1}}.$$

The Sobolev embedding theorem now implies that for $p > \frac{d}{2}$, $\|\mathbf{u}^h\|_{L^\infty} \leq C(\Omega)\|\nabla\mathbf{u}^h\|_{L^p}$ and the result follows. □

3. Error analysis for the Oseen problem. This section presents an error analysis of the SGS model discretized by the finite element method for a linearized Navier-Stokes problem. The error is calculated with respect to the solution of (1.1), unregularized. This analysis complements the finite element error analysis for the Ladyzhenskaya model (see Ladyzhenskaya [18]) of fluid flow in Du and Gunzburger [7-9] in that the former considers the nonlinear problem with the parameters μ and Re sufficiently small to ensure global uniqueness of solutions while herein we study $\mu = \mu(h) \to 0$ and Re large for a (linearized) Oseen problem. There is also other work on the pure (symmetric) p-Laplacian problem and a related creeping flow problem in Barrett and Liu [2, 3].

The Oseen problem is given as follows. For $\mathbf{U}(x, y)$ a smooth enough, known flow fluid with div $\mathbf{U} = 0$, we seek $\mathbf{u}(x, y)$ satisfying

$$\left\{\begin{array}{l} -\mathrm{Re}^{-1}\Delta\mathbf{u} + \mathbf{U}\cdot\nabla\mathbf{u} + \nabla\lambda = \mathbf{f} \text{ in } \Omega,\ \nabla\cdot\mathbf{u} = 0 \text{ in } \Omega, \\ \mathbf{u} = 0 \text{ on } \partial\Omega,\ \displaystyle\int_\Omega \lambda dx = 0. \end{array}\right. \tag{3.1}$$

Error analysis for (3.1) at all Reynolds numbers is possible using monotonicity techniques since the solution operator to (3.1) is monotone on the space

$$\mathbf{V} := \left\{\mathbf{v} \in \left(\overset{\circ}{H}{}^1(\Omega)\right)^d : \nabla\cdot\mathbf{v} = 0\right\}$$

of divergence-free functions, [12, 13].

The numerical method for (3.1) is as follows. Choose the velocity-pressure finite element spaces $(\mathbf{X}^h, Q^h)$ such that $\mathbf{X}^h \subset (\overset{\circ}{H}{}^1(\Omega))^d$, $Q^h \subset L_0^2(\Omega) := \{q \in L^2(\Omega) : \int_\Omega q dx = 0\}$ and they satisfy the inf-sup or Babuska-Brezzi condition:

$$\inf_{q^h\in Q^h} \sup_{v^h\in\mathbf{X}^h} \frac{\int_\Omega q^h \nabla\cdot\mathbf{v}^h dx}{\|q^h\|\ \|\nabla\mathbf{v}^h\|} = \beta^h > 0, \tag{3.2}$$

β^h being bounded away from zero uniformly in h. Examples of "good" spaces satisfying (3.2) are given in [12, 13]. The approximation is now defined by the following: seek $(\mathbf{u}^h, p^h) \in (\mathbf{X}^h, Q^h)$ satisfying for all $(\mathbf{v}, q) \in (\mathbf{X}^h, Q^h)$,

$$AV_p(\mathbf{u}^h, \mathbf{v}) + a(\mathbf{u}^h, \mathbf{v}) + b(\mathbf{U}, \mathbf{u}^h, \mathbf{v}) + (\lambda^h, \nabla\cdot\mathbf{v}) - (q, \nabla\cdot\mathbf{u}^h) = (\mathbf{f}, \mathbf{v}). \tag{3.3}$$

LEMMA 3.1 (max-norm stability). *Under the assumptions of Lemma* 2.2 *the solution* $\mathbf{u}^h$ *to* (3.3) *satisfies all the stability bounds of Lemma* 2.2. □

Since $AV_p(\cdot,\cdot)$ is associated with the "p-Laplacian," its monotonicity properties are documented in many places; see e.g. [2-4, 7-9, 24, 30]. These properties will now be summarized here; see the above references for proofs. Using the Riesz representation theorem, define the operator $T : \mathbf{X} \to \mathbf{X}'$ satisfying

$$(T(\mathbf{u}), \mathbf{v}) = (|\nabla\mathbf{u}|^{p-2}\nabla\mathbf{u}, \nabla\mathbf{v}) \quad \forall\mathbf{v} \in \mathbf{X}.$$

PROPOSITION 3.1. *Let* $p \geq 2$. *Then* T *satisfies for all* $\mathbf{u}, \mathbf{v} \in (\overset{\circ}{W}{}^{1,p}(\Omega))^d$,

$$(T(\mathbf{u}) - T(\mathbf{v}), \mathbf{u} - \mathbf{v}) \geq \zeta(\|\nabla(\mathbf{u} - \mathbf{v})\|_{L^p})\|\nabla(\mathbf{u} - \mathbf{v})\|_{L^p},$$

$$\|T(\mathbf{u}) - T(\mathbf{v})\|_{W^{-1,q}} \leq \Gamma(r)\|\nabla(\mathbf{u} - \mathbf{v})\|_{L^p},$$

for $\|\nabla\mathbf{u}\|_{L^p} \leq r$ *and* $\|\nabla\mathbf{v}\|_{L^p} \leq r$, *where* $\Gamma(r) = C(2p - 3)r^{p-2}$, *and* $\zeta(s) = C(\frac{1}{2})^{p-2}s^{p-1}$.

Proof. These bounds have been proven, for example, in [2, 3, 7-10, 24, 30]. □

The following is an easy consequence of strong monotonicity, [30], of T and the inf-sup condition.

PROPOSITION 3.2. *Let $p \geq 2$ and suppose $(\mathbf{X}^h, Q^h)$ satisfies* (3.2). *Then,*

(i) *The subspace $\mathbf{V}^h$ of discretely divergence-free functions*

$$\mathbf{V}^h := \left\{\mathbf{v}^h \in \mathbf{X}^h : (\mathbf{v}^h, q^h) = 0 \, \forall \, q^h \in Q^h\right\}$$

is well defined and nontrivial. Further, for $\mathbf{u} \in \mathbf{X}$ satisfying $\operatorname{div}(\mathbf{u}) = 0$,

$$\inf_{\mathbf{v} \in \mathbf{V}^h} \|\nabla(\mathbf{u} - \mathbf{v})\| \leq C\left(1 + \frac{1}{\beta_h}\right) \inf_{\mathbf{v} \in \mathbf{X}^h} \|\nabla(\mathbf{u} - \mathbf{v})\|.$$

(ii) *There is a well-defined continuous mapping $T_p^h : \mathbf{V}^{h'} \to \mathbf{V}^h$ satisfying*

$$(T_p^h(\mathbf{u}), \mathbf{v}) = A V_p(\mathbf{u}, \mathbf{v}) + \mathrm{Re}^{-1}(\boldsymbol{\nabla}\mathbf{u}, \nabla\mathbf{v}) + b(\mathbf{U}, \mathbf{u}, \mathbf{v}).$$

(iii) *$T_p^h(\cdot)$ satisfies the following: for all $\mathbf{u}, \mathbf{v} \in \mathbf{V}^h$,*

$$(T_p^h(\mathbf{u}) - T_p^h(\mathbf{v}), \mathbf{u}-\mathbf{v}) \geq C\mu(h)\zeta(\|\nabla(\mathbf{u}-\mathbf{v})\|_{L^p})\|\nabla(\mathbf{u}-\mathbf{v})\|_{L^p} + \mathrm{Re}^{-1}\|\nabla(\mathbf{u}-\mathbf{v})\|_{L^2}^2,$$

where $\zeta(\cdot)$ is given in Proposition 3.1.

Proof. Part (i) is from standard finite element convergence theory for the linear Stokes problem; see, for example, [12, 13] for a proof. Part (ii) follows immediately from the Riesz representation theorem. Part (iii) follows from Proposition 3.1 and the skew symmetry of $b(\cdot, \cdot, \cdot)$. □

COROLLARY 3.1. *Suppose $p \geq 2$ and* (3.3) *holds. The solution $(\mathbf{u}^h, \lambda^h)$ to* (3.1) *exists uniquely.*

Proof. Existence of $\mathbf{u}^h$ is equivalent to the solvability of $T_p^h(\mathbf{u}^h) = P^h f$ in $\mathbf{V}^h$ where P^h is the L^2 orthogonal projection $P^h : \mathbf{X}^h \to \mathbf{V}^h$. Existence and uniqueness of $T_p^h(\mathbf{u}^h) = P^h\mathbf{f}^h$ follows from Proposition 3.2 and Minty's lemma; see Minty [26]. Existence and uniqueness of λ^h then follows from the inf-sup condition exactly as for the Stokes problem; see [12, 13] for these arguments. □

THEOREM 3.1. *Suppose $(\mathbf{X}^h, Q^h)$ satisfies the inf-sup condition with constant β^h, and $\mathbf{X}^h$ satisfies the inverse estimates* (2.2) *and* (2.3). *Then, with $r = \|\nabla\mathbf{u}\|_{L^p}$,*

$$\begin{aligned}
&\mu(h)\| \nabla(\mathbf{u} - \mathbf{u}^h)\|_{L^p}^p + \mathrm{Re}^{-1}\|\nabla(\mathbf{u} - \mathbf{u}^h)\|^2 \\
&\quad\leq C\left(1 + \frac{1}{\beta_h}\right) \inf_{\mathbf{w} \in \mathbf{X}^h} \Big\{C(\mathbf{U})\|\mathbf{u} - \mathbf{w}\|^2 + \mathrm{Re}^{-1}\|\nabla(\mathbf{u} - \mathbf{w})\|^2 \\
&\qquad + \mu(h)\|\nabla(\mathbf{u} - \mathbf{w})\|_{L^p}^p + h^{d\left(\frac{2-p}{p}\right)}\mu(h)^2\|\nabla(\mathbf{u} - \mathbf{w})\|_{L^p}^2\Big\} \\
&\qquad + C\inf_{q \in Q^h} \|\lambda - q\|^2 + C\mathrm{Re}h^{d\left(\frac{2-p}{p}\right)}\mu(h)^2\|\nabla\mathbf{u}\|_{L^p}^{2p-2}.
\end{aligned} \tag{3.4}$$

Proof. The error bound is proven using Galerkin orthogonality, the inf-sup condition, and monotonicity of $T_p^h(\cdot)$ on $\mathbf{V}^h$. First the error equation is derived. Let $\mathbf{v} \in \mathbf{V}^h$ be arbitrary, then $\mathbf{u} \in \mathbf{X}$ satisfies

$$a(\mathbf{u}, \mathbf{v}) + A V_p(\mathbf{u}, \mathbf{v}) + b(\mathbf{U}; \mathbf{u}, \mathbf{v}) = (f, \mathbf{v}) - (\lambda - q^h, \nabla \cdot \mathbf{v}) + A V_p(\mathbf{u}, \mathbf{v})$$

for all $q^h \in Q^h$. Thus, with $\mathbf{e} = \mathbf{u} - \mathbf{u}^h$, for all $\mathbf{v} \in \mathbf{V}^h$,

$$\begin{aligned} AV_p(\mathbf{u}, \mathbf{v}) &- AV_p(\mathbf{u}^h, \mathbf{v}) + a(\mathbf{e}, \mathbf{v}) + b(\mathbf{U}, \mathbf{e}, \mathbf{v}) \\ &= AV(\mathbf{u}, \mathbf{v}) - (\lambda - q^h, \nabla \cdot \mathbf{v}). \end{aligned}$$

Let $\mathbf{w} \in \mathbf{V}^h$ be arbitrary and define $\phi = (\mathbf{w} - \mathbf{u}^h) \in \mathbf{V}^h$, $\eta = \mathbf{u} - \mathbf{w}^h$. Adding and subtracting terms as appropriate gives

$$\begin{aligned} AV_p(\mathbf{w}, \mathbf{v}) &- AV_p(\mathbf{u}^h, \mathbf{v}) + a(\phi, \mathbf{v}) + b(\mathbf{U}, \phi, \mathbf{v}) \\ &= -(\lambda - q^h, \nabla \cdot \mathbf{v}) - b(\mathbf{U}, \eta, \mathbf{v}) \\ &\quad -a(\eta, \mathbf{v}) + AV_p(\mathbf{w}, \mathbf{v}) - AV_p(\mathbf{u}, \mathbf{v}) + AV_p(\mathbf{u}, \mathbf{v}). \end{aligned}$$

Equivalently, for all $\mathbf{v} \in \mathbf{V}^h$,

$$\begin{aligned} (T_p^h(\mathbf{w}), \mathbf{v}) - (T_p^h(\mathbf{u}^h), \mathbf{v}) &= -(\lambda - q^h, \nabla \cdot \mathbf{v}) \\ &\quad -b(\mathbf{U}, \boldsymbol{\eta}, \mathbf{v}) - a(\boldsymbol{\eta}, \mathbf{v}) + AV_p(\mathbf{u}, \mathbf{v}) \\ &\quad +AV_p(\mathbf{u}, \mathbf{v}) - AV_p(\mathbf{w}, \mathbf{v}). \end{aligned}$$

Setting $\mathbf{v} = \phi$ (since $\phi \in \mathbf{V}^h$), using strong monotonicity on the left-hand side of the above equation and local-Lipschitz continuity on the right-hand side gives

$$\begin{aligned} C\mu\ (h)\zeta(\|\nabla\phi\|_{L^p})\|\nabla\phi\|_{L^p} &+ \mathrm{Re}^{-1}\|\nabla\phi\|^2 \\ \leq \|\lambda - q\|\,\|\nabla\phi\| &+ C(\mathbf{U})\|\boldsymbol{\eta}\|\,\|\nabla\phi\| \\ +\mathrm{Re}^{-1}\|\nabla\boldsymbol{\eta}\|\ \|\nabla\phi\| &+ AV_p(\mathbf{u}, \phi) \\ +\mu(h)\Gamma(r_1)\|\nabla(\mathbf{u} - \mathbf{w})\|_{L^p}&\|\nabla\mathbf{w}\|_{L^p}, \end{aligned}$$

where $r_1 = \max\{\|\nabla\mathbf{u}\|_{L^p},\ \|\nabla\mathbf{w}\|_{L^p}\}$.

Now $AV_p(\mathbf{u}, \phi) = \mu(h)(T(\mathbf{u}) - T(0), \phi) \leq \mu(h)\|T(\mathbf{u}) - T(0)\|_{W^{-1,q}}\|\phi\|_{W^{-1,p}} \leq \mu(h)$ $\Gamma(r_2)\|\nabla\mathbf{u}\|_{L^p}\|\nabla\phi\|_{L^p}$, with $r_2 = \|\mathbf{u}\|_{W^{1,p}}$. Thus,

$$AV_p(\mathbf{u}, \phi) \leq \mu(h)\Gamma(r_2)\|\nabla\mathbf{u}\|_{L^p}^{p-1}\|\nabla\phi\|_{L^p}.$$

Thus,

$$\begin{aligned} C\ \mu(h)\zeta(\|\nabla\phi\|_{L^p})\|\nabla\phi\|_{L^p} + C\mathrm{Re}^{-1}\|\nabla\phi\|^2 &\leq \|\lambda - q\|^2 + C(\mathbf{U})\|\boldsymbol{\eta}\|^2 \\ +\mathrm{Re}^{-1}\|\nabla\boldsymbol{\eta}\|^2 + \mu(h)\Gamma(r_2)\|\nabla\mathbf{u}\|_{L^p}^{p-1}\|\nabla\phi\|_{L^p} &+ \mu(h)\Gamma(r_1)\|\nabla\boldsymbol{\eta}\|_{L^p}\|\nabla\phi\|_{L^p}. \end{aligned}$$

To complete the error analysis note that since $\phi \in \mathbf{X}^h$, ϕ satisfies the inverse estimate (2.3). Indeed, $\|\nabla\phi\|_{L^p} \leq \gamma(h)\|\nabla\phi\|$ for all $\phi \in \mathbf{X}^h$, where $\gamma(h) = Ch^{\frac{d}{2}((2-p)/p)}$, so that

$$\begin{aligned} C\mu(h)\zeta(\|\nabla\phi\|_{L^p})\|\nabla\phi\|_{L^p} + C\mathrm{Re}^{-1}\|\nabla\phi\|^2 &\leq \|\lambda - q\|^2 + C(\mathbf{U})\|\boldsymbol{\eta}\|^2 \\ +\mathrm{Re}^{-1}\|\nabla\boldsymbol{\eta}\|^2 &+ C\mathrm{Re}\gamma^2(h, p)\mu(h)^2\Gamma^2(r_2)\|\nabla\mathbf{u}\|_{L^p}^{2p-2} \\ &+ C\mathrm{Re}\Gamma^2(r_1)\gamma^2(h)\mu(h)^2\|\nabla\boldsymbol{\eta}\|_{L^p}^2. \end{aligned}$$

Under the inf-sup condition an infimum over $\mathbf{V}^h$ may be replaced by a constant times an infimum over $\mathbf{X}^h$. Thus the result follows from the triangle inequality, the fact that at the infimum $r_1, r_2 \leq Cr_0$, and (since $\|\mathbf{f} + g\|_{L^p}^p \leq C(p)(\|\mathbf{f}\|_{L^p}^p + \|g\|_{L^p}^p)$)

$$\begin{aligned} \zeta(\|\nabla(\mathbf{u} - \mathbf{u}^h)\|_{L^p})\ &\|\nabla(\mathbf{u} - \mathbf{u}^h)\|_{L^p} \\ &\leq C\zeta(\|\nabla\phi\|_{L^p})\|\nabla\phi\|_{L^p} + 2C\zeta(\|\nabla\boldsymbol{\eta}\|_{L^p})\|\nabla\boldsymbol{\eta}\|_{L^p}. \qquad \square \end{aligned}$$

It is interesting to compare (3.4) with the usual error estimate for the (centered) Galerkin method. With a careful error analysis, the usual Galerkin approximation of the Oseen problem satisfies the error bound:

$$\begin{aligned}\mathrm{Re}^{-1}\|\nabla(\mathbf{u}-\mathbf{u}^h)\|^2 &\le C(1+\beta_h^{-1})\inf_{\mathbf{w}\in\mathbf{X}^h}\left\{\mathrm{Re}^{-1}\|\nabla(\mathbf{u}-\mathbf{w})\|^2+\|\mathbf{u}-\mathbf{w}\|^2\right\}\\ &\quad + C\mathrm{Re}\inf_{q\in Q^h}\|\lambda-q\|^2.\end{aligned} \tag{3.5}$$

Suppose, for example, the "Mini" element [13, p. 36] is used (equal order interpolation is optimal if $\mathrm{Re}^{-1} \le O(h)$, [19, 21]); then (3.5) predicts an error bound of the rough form $\|\nabla(\mathbf{u}-\mathbf{u}^h)\| \le C_1(\mathbf{u},\lambda)(h^2+\mathrm{Re}^2h^4)$, which is no better than $O(1)$ when $\mathrm{Re} = O(h^{-2})$. With the same reasoning (ignoring logarithm terms), the error bound (3.4) roughly states that (fixing $d = 3$, $\mu(h) = \mu_0 h^\sigma$, $p = \sigma + 1$) $h^\sigma\|\nabla(\mathbf{u}-\mathbf{u}^h)\|_{L^p}^p + \mathrm{Re}^{-1}\|\nabla(\mathbf{u}-\mathbf{u}^h)\|^2 \le C_2(\mathbf{u},\lambda)\left\{h^4+\mathrm{Re}^{-1}h^2+h^{2\sigma+1}+\mathrm{Re}h^{2\sigma-3+3/(\sigma+1)}\right\}$. For example, if $\mathrm{Re} = O(h^{-2})$ and $\sigma = 4$, this predicts $\|\nabla(\mathbf{u}-\mathbf{u}^h)\|^2 \le C_2(\mathbf{u},\lambda)h^{8/5}$.

Naturally the constants $C_{1,2}(\mathbf{u}, p)$ in the above rough estimates depend upon Re implicitly due to local fluid behavior. The centered scheme is well known to do a poor job localizing the errors in such cases. Approximate solutions are globally oscillating due to local solution effects in small regions. Hopefully, the formulations (2.1), (3.3) introduce sufficient viscosity in such regions to better localize the errors due to large derivatives of $\mathbf{u}$ and p. There is some experimental evidence of this in §4; see Figures 4.1-4.3.

4. An illustration. Due to the complexity of solutions to the Navier-Stokes equations and the ambiguity of the notion of "solution quality," it has become customary to test discretizations tailored to higher Reynolds number problems first upon singularly perturbed convection diffusion equations such as the following: seek $\phi(x, y)$ satisfying

$$\begin{aligned}&-\epsilon\Delta\phi + \mathbf{U}\cdot\nabla\phi + 2\phi = f \text{ in } \Omega, \quad \phi = \alpha \text{ on } \partial\Omega,\\ &0 < \epsilon << O(h), \ \nabla\cdot\mathbf{U} = 0 \text{ in } \Omega.\end{aligned} \tag{4.1}$$

Equation (4.1) satisfies a maximum principle (unlike the Navier-Stokes equations) and, due to the additional "$+2\phi$" term, a fairly complete asymptotic picture of solutions is available. Therefore, at least *lack* of "solution quality" is clearcut for (4.1)! Further, (4.1) also arises coupled to the Navier-Stokes equations as an energy equation [5], so it is interesting in its own right.

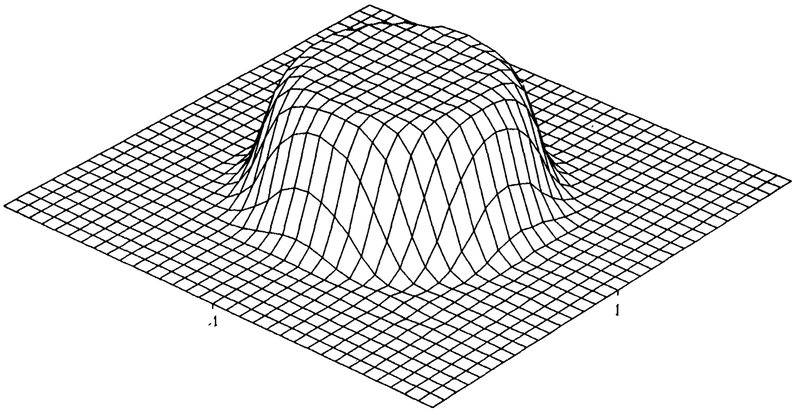


FIG. 4.1. *Approximate solution,* $\mu_0 = 1$, $p = 3$, $\sigma = 2$ *using linear elements.*

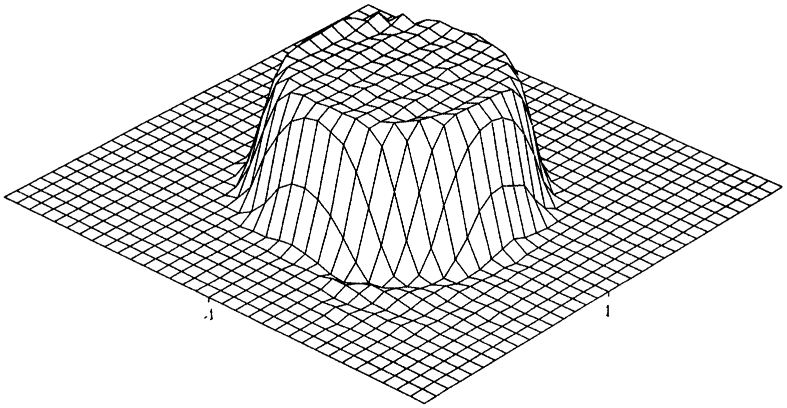


FIG. 4.2. *Approximate solution, $\mu_0 = 1$, $p = 5$, $\sigma = 3$, using linear elements.*

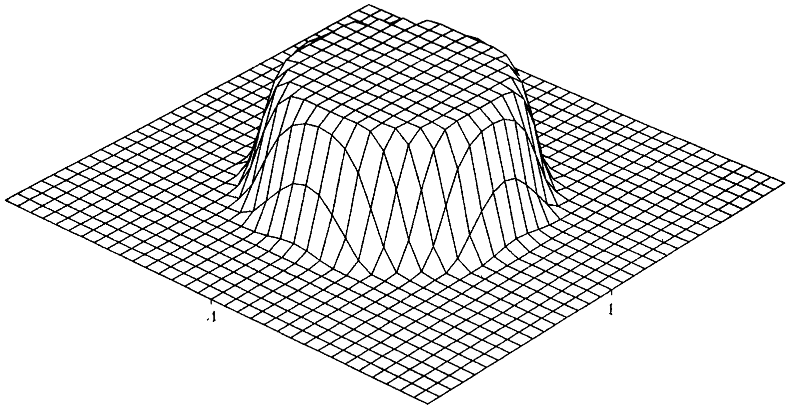


FIG. 4.3. *The (Linear interpolant of) an approximate solution, $\epsilon = .1h^2$, $h = \frac{1}{32}$ using $\mu_0 = 1$, $p = 5$, $\sigma = 3$, quadratic elements.*

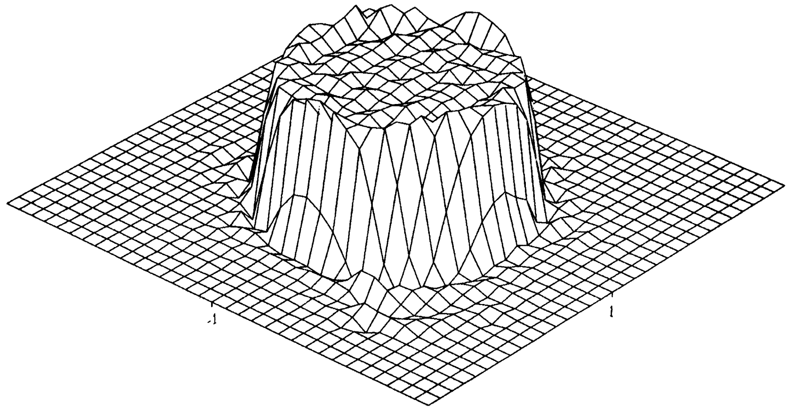


FIG. 4.4. *Usual FEM approximation, without nonlinear SGS terms.*

To illustrate the effect of the additional nonlinear artificial viscosity term upon "solution quality," we consider a rotating pulse problem of the form (4.1) approximated with linear and quadratic elements. The usual centered Galerkin formulation plus nonlinear artificial viscosity $AV_p(\cdot, \cdot)$ is used.

Example (internal flow, rotating pulse problem). We have

$$\Omega = (-1, 1) \times (-1, 1), \quad \mathbf{U} = (-yR(r),\ xR(r)),$$

where $r^2 = x^2 + y^2$, $R(r) = 0$ for $r \geq 1$ is a nonnegative C^1 function with $R(r) > 0$ for $r < 1$, $R(0) = 1$, so that $\nabla \cdot \mathbf{U} = 0$. We choose $\epsilon = .1h^2$, and $f \equiv 1$ if $r \leq \frac{1}{2}$, $f \equiv 0$ if $r > \frac{1}{2}$. This emulates a rotating flow inside a circular enclosure ($r \leq 1$). The true solution for ϵ small is a rotating "blob" or circular table with an $O(\sqrt{\epsilon})$ transition layer, $\phi \sim \frac{1}{2}$ if $r < \frac{1}{2}$ and $\phi \sim 0$ if $r > \frac{1}{2}$.

The same mesh was used for all tests: a uniform rectilinear mesh $h = \frac{1}{32}$, each square divided into two triangles by a line of slope -1. This mesh is *not* adapted to the circular symmetry of the convection field. The nonlinear equations were solved using a damped Newton method, and the linearizations, were solved using the conjugate gradient squared (CGS) method using an ILU_0 preconditioner; see Maubach [25] for a presentation of these methods. With a nonzero initial guess no difficulty was encountered. The only difficulty seen was in solving the linearized system arising from a zero initial guess. Specifically, with a zero initial guess for Newton's method, the first iterate is the solution of the usual (centered) Galerkin linear system. This *linear* system is much more sensitive and difficult to solve using standard preconditioners than all the remaining linear and nonlinear problems. Aside from using a nonzero initial guess or a better preconditioner, e.g., [22], another attractive possibility is to begin with a linear artificial viscosity approximation and "antidiffuse" using a few defect correction steps, where the $AV_p(\cdot, \cdot)$-nonlinearity only occurs in the residual calculation. This requires solving only a few (stable) linear systems with the same coefficient matrix. This combination has not yet been tested, but the practical success and theoretical support for this type of procedure in other similar contexts [1, 19] and Hemker and Koren [15, 16] suggest adapting it to the present one as well.

In the tests depicted in Figures 4.1-4.3, note the good quality of the approximate solutions (compare Figures 4.1, 4.2, and 4.3 to Figure 4.4) and the capturing of the transition regions in about $3h$ distance. With optimized parameter selections this could likely be further sharpened.

Acknowledgment. I thank J. Maubach for generously providing the examples in §4.

REFERENCES

[1] O. Axelsson and W. Layton, *Defect correction methods for convection dominated, convection diffusion equations*, RAIRO Modél. Math. Anal. Numér. 24 (1990), pp. 423-455.

[2] J. W. Barrett and W. B. Liu, *Finite element approximation of the p-Laplacian*, Math. Comp., 61 (1993), pp. 523-538.

[3] ———, *Finite element error analysis of a quasi-Newtonian flow obeying the Carreau or power law*, Numer. Math., 64 (1993), pp. 433-453.

[4] R. E. Boisvert and W. F. Ames, *Group properties and new solutions of the Navier-Stokes equations*, J. Engrg. Math., 17 (1983), pp. 203-221.

[5] J. Boland and W. Layton, *Error analysis for finite element methods for steady natural convection problems*, Numer. Funct. Anal. Optim., 11 (1990), pp. 449-483.

[6] L. Caffarelli, R. Kohn, and L. Nirenberg, *Partial regularity of suitable weak solutions of the Navier-Stokes equations*, Comm. Pure Appl. Math., 25 (1982), pp. 791-831.

[7] Q. Du and M. Gunzburger, *Analysis of a Ladyzhenskaya model for incompressible viscous flow*, J. Math. Anal. Appl., 155 (1991), pp. 21-45.

[8] ———, *Finite element approximation for a Ladyzhenskaya model for stationary incompressible viscous flows*, SIAM J. Numer. Anal., 27 (1990), pp. 1-19.

[9] Q. Du, M. Gunzburger, and L. S. Hou, *Analysis and finite element approximation of optimal control problems for a Ladyzhenskaya model for stationary, incompressible, viscous flows*, submitted (1993).

[10] T. Dubois, F. Jauberteau, and R. Temam, *Subgrid modelling and the interaction of small and large wavelengths in turbulent flows*, Comput. Phys. Comm., 65 (1991), pp. 100–106.

[11] T. Dupont and L. R. Scott, *Polynomial approximation of functions in Sobolev spaces*, Math. Comp., 34 (1980), pp. 441–463.

[12] V. Girault and P. A. Raviart, *Finite Element Methods for the Navier–Stokes Equations: Theory and Algorithms*, Springer-Verlag, Berlin, 1986.

[13] M. Gunzburger, *Finite Element Methods for Viscous Incompressible Flow: A Guide to Theory, Practice and Algorithms*, Academic Press, Boston, 1989.

[14] M. Gunzburger and J. Turner, *An analysis of approximation of an algebraic model of turbulence*, Comput. Math. Appl., 15 (1988), pp. 945–951.

[15] P. W. Hemker, *An accurate method without directional bias for the numerical solution of a 2-D elliptic singular perturbation problem*, in Thy. and Appls. of Sing. Perts. 942, W. Eckhaus and E. M. Jaeger, eds., Springer LNM, Berlin, 1982.

[16] P. W. Hemker and B. Koren, *Multigrid defect correction and upwind schemes for the steady Euler equations*, in Numer. Methods for Fluid Dynamics III, K. W. Morton and M. J. Bainer, eds., Clarendon, Oxford, 1988, pp. 153–170.

[17] J. G. Heywood and R. Rannacher, *On the question of turbulence modelling by approximate inertial manifolds and the nonlinear Galerkin method*, SIAM J. Numer. Anal., 30 (1993), pp. 1603–1621.

[18] O. A. Ladyzhenskaya, *Modification of the Navier–Stokes equations for large velocity gradients*, Boundary Value Problems of Mathematical Physics and Related Aspects of Function Theory, Consultants Bureau, New York, 1970.

[19] W. Layton, *Solution algorithms for incompressible viscous flows at high Reynolds number*, Vestnik Moskov. Gos. Univ. (Computational Math. and Cybernetics), series 15, No. 4 (1994).

[20] ———, *A two level discretization method for the Navier–Stokes equations*, Comp. Math. Appl., 26 (1993), pp. 33–38.

[21] W. Layton and W. Lenferink, *A multi-level mesh independence principle for the Navier–Stokes equations*, SIAM J. Numer. Anal., 33 (1996), pp. 17–30.

[22] W. Layton, J. Maubach, and P. Rabier, *Robustness of an Elementwise Parallel Iterative Method for Convection-Diffusion Problems*, I. C. M. A. Report, Univ. of Pittsburgh, 1993.

[23] D. C. Leslie and G. L. Quarini, *The application of turbulence theory to the formulation of subgrid modelling procedures*, J. Fluid Mech., 91 (1979), pp. 65–91.

[24] J. Manfredi, *Regularity of minima for functionals with p-growth*, J. Differential Equations, 76 (1988), pp. 203–212.

[25] J. Maubach, *Iterative Methods for Nonlinear Partial Differential Equations*, C.W.I. Press, Amsterdam, 1991.

[26] G. Minty, *Monotone (nonlinear) operators in Hilbert space*, Duke Math. J., 29 (1962), pp. 341–346.

[27] R.V. Southwell, *Stress calculation in frameworks by the method of systematic relaxation of constraints*, I, II, Proc. Roy. Soc. London Ser. A., 15 (1935), pp. 56–95.

[28] L. Tobiska, *Conforming and Nonconforming Finite Element Methods of Streamline Diffusion Type for Solving the Navier–Stokes Equations*, Report 12/90, Tech. Univ. "Otto von Guericke," Magdeburg, 1990.

[29] L. Tobiska and R. Verfürth, *Analysis of a streamline diffusion finite element method for the Stokes and Navier–Stokes equations*, Preprint 1/92, Tech. Univ. "Otto von Guericke," Magdeburg, 1992.

[30] M. M. Vainberg, *Variational Methods for the Study of Nonlinear Operators*, Holden-Day, San Francisco, CA, 1964.

[31] J. Xu, *Two-grid finite discretization techniques for linear and nonlinear PDE*, SIAM J. Numer. Anal., 33 (1996), to appear.

SIAM J. SCI. COMPUT.
Vol. 17, No. 2, pp. 358-369, March 1996

A NUMERICAL METHOD FOR THE INCOMPRESSIBLE NAVIER-STOKES EQUATIONS BASED ON AN APPROXIMATE PROJECTION*

ANN S. ALMGREN†, JOHN B. BELL†, AND WILLIAM G. SZYMCZAK‡

Abstract. In this method we present a fractional step discretization of the time-dependent incompressible Navier-Stokes equations. The method is based on a projection formulation in which we first solve diffusion-convection equations to predict intermediate velocities, which are then projected onto the space of divergence-free vector fields. Our treatment of the diffusion-convection step uses a specialized second-order upwind method for differencing the nonlinear convective terms that provides a robust treatment of these terms at a high Reynolds number. In contrast to conventional projection-type discretizations that impose a discrete form of the divergence-free constraint, we only approximately impose the constraint; i.e., the velocity field we compute is not exactly divergence-free. The approximate projection is computed using a conventional discretization of the Laplacian and the resulting linear system is solved using conventional multigrid methods. Numerical examples are presented to validate the second-order convergence of the method for Euler, finite Reynolds number, and Stokes flow. A second example illustrating the behavior of the algorithm on an unstable shear layer is also presented.

Key words. incompressible flow, projection methods

AMS subject classifications. 65-C20, 76-D05

1. Introduction.

In this paper we develop a projection method for the incompressible Navier-Stokes equations

$$U_t + (U \cdot \nabla)U = \varepsilon \Delta U - \nabla p + F, \tag{1.1}$$

$$\nabla \cdot U = 0, \tag{1.2}$$

which is formally second-order accurate. Here U represents the velocity field, p represents the hydrodynamic pressure, and F represents any external forces. We denote the x and y components of velocity by u and v, respectively. The method presented here is based on the algorithms presented by Bell, Colella, and Glaz [1] and Bell, Colella, and Howell [2]. As in those papers, we use a second-order upwind method for the treatment of the nonlinear convective terms in (1.1). The algorithms presented in those papers were motivated by a desire to apply higher-order upwind methods developed for inviscid, compressible flow to the incompressible Navier-Stokes equations. In particular, they used a specialized version of the unsplit second-order Godunov methodology introduced for gas dynamics by Colella [9]. The upwind methodology provides a robust discretization of the convective terms that avoids any cell-Reynolds-number stability restriction for high Reynolds-number flow.

The focus of this paper is on the discretization of the projection. The original projection method developed by Chorin [6] defines the projection by defining discrete operators D and G, approximating divergence and gradient, which are skew adjoint; i.e.,

$$D = -G^T.$$

With this definition the discrete projection

$$\mathbf{P} = I - G(DG)^{-1}D$$

*Received by the editors February 8, 1993; accepted for publication (in revised form) March 29, 1995.

†The work of these authors was performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under contract W-7405-Eng-48. Support under contract W-7405-Eng-48 was provided by the Applied Mathematical Sciences Program of the Office of Energy Research and by the Defense Nuclear Agency under IACRO 92-825 and IACRO 93-817. Lawrence Livermore National Laboratory, Livermore, CA 94550 (almgren@.llnl.gov, gbb@llnl.gov).

‡Naval Surface Warfare Center, Silver Spring, MD 20903 (wszymcz@nswc-wo.navy.mil).

(with boundary conditions implicit in the boundary conditions of the flow problem) is a discrete orthogonal projection on the finite-dimensional space of vector fields defined on the mesh. In Chorin's formulation both pressure and velocity are specified at nodes and central differences are used for the definition of D and G. This results in an expanded five-point stencil for the discrete Laplacian, DG, that must be inverted to apply the projection. This expanded stencil produces a local decoupling of the mesh points with a 2^d-dimensional kernel for G where d is the dimension of the problem. Bell, Colella, and Glaz [1] use a discretization of the projection based on a finite element method due to Fortin [10] that uses pressure defined on cell centers with velocities given at nodes. This approach produces a more compact stencil but also generates a local decoupling of the grid and dim ker $G > 1$. Bell, Colella, and Howell [2] use a fully cell-centered analogue of Chorin's algorithm. This scheme exhibits a local decoupling, but in the presence of Dirichlet boundary conditions the cell-centered approximation eliminates the nonconstant elements in ker G.

Although the projections discussed above have been shown to be effective for solution of the incompressible Navier-Stokes equations, they exhibit a number of shortcomings as a result of the local grid decoupling. First, the schemes all use nonstandard discretizations of DG that require specialized iterative procedures that properly respect the stencil that is used. (See [2, 13] for a discussion of such a procedure.) For the schemes in which dim ker $G > 1$, the nonconstant elements in the kernel induce additional, artificial compatibility constraints similar to the physical condition

$$\int_{\partial\Omega} U \cdot \mathbf{n}\, ds = 0$$

on the boundary of the physical domain Ω. Local decoupling raises additional difficulties when more complex low Mach number processes are being modeled. Often in these cases a term that results from the additional physics, and is not in the range of the divergence operator, is added to the right-hand side of the Poisson equation for pressure. The presence of the additional source term can result in marked oscillations in the solution. Lai [16, 15] reports such oscillations when using projections with locally decoupled stencils for modeling low Mach number combustion. The approximate projection introduced here has successfully been used for this type of simulation; see [19].

The goal of the work reported here is to develop a projection that avoids the difficulties associated with locally decoupled stencils and is amenable to treatment by standard iterative methods. Several approaches have been proposed that address this issue. In a MAC discretization (cf. Harlow and Welch [11]), a discrete form of (1.2) is enforced in a context in which DG is the standard five-point finite difference Laplacian. However, in the MAC approach the components of velocity are not colocated, which makes the application of modern higher-order upwind techniques extremely cumbersome if not impossible. Strikwerda [20] proposes a regularized projection based on Chorin's original node-based algorithm. Strikwerda's method uses a skewed, higher-order third derivative perturbation to D and G to remove the local decoupling of Chorin's method. Colella [7] reports that although Strikwerda's algorithm is effective for homogeneous boundary conditions, noticeable anomalies arise from the asymmetry of the operators when inflow and outflow profiles are specified. Although the overall algorithm can be engineered around this shortcoming, the complexity of the method is considerably enhanced. In addition, the Strikwerda projection also generates a wide stencil requiring care in the treatment of boundary conditions.

We would like to develop a form of the projection compatible with a cell-centered discretization of velocity that avoids any local decoupling of the stencils, provides a symmetric discretization of the potential flow inherent in nonhomogeneous boundary conditions, and generates a linear system that fits the framework of conventional fast iterative techniques (e.g., multigrid) for second-order elliptic equations. The evidence to date suggests that this is not possible while simultaneously requiring that there is a discrete divergence D so that $DU = 0$ is satisfied. In this paper we will relax that condition and only require that $DU = O(h^2)$, where h is the mesh spacing. The approach will treat velocities as averages over grid cells and discretize pressure using a standard bilinear finite element discretization. We will show that relaxing the treatment of (1.2) does not have any deleterious effects on the stability or convergence properties of the algorithm; the advantages of the higher-order upwind methodology are retained and the complexity of the linear algebraic aspects of the method is substantially reduced.

In the next two sections we will review the basic fractional step scheme of Bell, Colella, and Glaz [1] and describe the discretization of the convection-diffusion step of the algorithm. The fourth section describes the approximation of the projection and briefly describes a multigrid algorithm for its solution. This section also contains a discussion of boundary conditions for the projection. In the fifth section we present computational results obtained with the method. The first sequence of results demonstrates second-order convergence of the algorithm for finite Reynolds number, Euler, and Stokes flows. The final example illustrates the application of the method to the computation of an unstable shear layer. The sixth section contains the conclusions.

2. Fractional step scheme. In this section we review the basic fractional step scheme used in the algorithm. In the presentation of the basic algorithm we will focus on homogeneous Dirichlet boundary conditions and no external forces. We will also assume that the mesh spacing on the base grid is uniform in the x- and y-directions, with $\Delta x = \Delta y = h$. These restrictions are not inherent limitations of the method; they have been adopted here for clarity of exposition. The reader is referred to [1, 4] for a more detailed description.

Our strategy for solving the system (1.1)-(1.2) is a fractional step scheme that has two parts: first we solve the advection-diffusion equations (1.1) without strictly enforcing the incompressibility constraint. Then, we project an intermediate vector field onto the space of discretely divergence-free vector fields.

For the diffusion-convection step we solve

$$\frac{U^* - U^n}{\Delta t} + [(U \cdot \nabla)U]^{n+\frac{1}{2}} = \varepsilon \Delta^h \left(\frac{U^n + U^*}{2} \right) - \nabla p^{n-\frac{1}{2}} \tag{2.1}$$

for the intermediate velocity U^*, where Δ^h is a second-order finite difference approximation to Δ. The pressure gradient is evaluated at $t^{n-1/2}$ and is treated as a source term in (2.1). (The pressure gradient is only computed at the 1/2-time levels.) The advection terms in (2.1), namely $[(U \cdot \nabla)U]^{n+1/2}$, are approximated at time $t^{n+1/2}$ to second-order in space and time using an explicit predictor-corrector scheme. This scheme uses only the available data at t^n; thus, the implicit part of (2.1) corresponds to two decoupled parabolic equation solves.

The velocity field U^* computed in the first step is not, in general, divergence-free. The projection step of the algorithm decomposes the result of the first step into a discrete gradient of a scalar potential and a discretely divergence-free vector field that correspond, respectively, to the new approximation to the pressure gradient and an update for the velocity. In particular, if $\mathbf{P}$ represents the composite grid projection then

$$\frac{U^{n+1} - U^n}{\Delta t} = \mathbf{P} \left(\frac{U^* - U^n}{\Delta t} \right), \tag{2.2}$$

$$\nabla p^{n+\frac{1}{2}} = \nabla p^{n-\frac{1}{2}} + (\mathbf{I} - \mathbf{P}) \left(\frac{U^* - U^n}{\Delta t} \right).$$

(Note that the vector field we project is not U^*; it is an approximation of U_t.)

3. Discretization of the diffusion-convection step. In this section we describe the algorithm for the diffusion-convection step in the fractional step scheme, namely (2.1). The algorithm is essentially the same as the second-order upwind method used by Bell, Colella, and Glaz [1]. Here, U^n_{ij} represents the value of the velocity field in cell B_{ij} at time t^n, and $Gp^{n-1/2}_{ij}$ represents the average value of ∇p in cell B_{ij} at time $t^{n-1/2}$. We note that although for the purposes of the projection the pressure is a bilinear function on each cell, we do not use that structure in the advection step. The algorithm is a predictor-corrector method. In the predictor, characteristic equations are used to extrapolate velocities to cell edges at the new half-time level $t^{n+1/2}$. In the corrector, the predicted values are used to compute a flux, which is then differenced to compute the advection terms. These two steps are described in more detail below.

Predictor. In the predictor we extrapolate the velocity and density to the cell edges at $t^{n+1/2}$ using a second-order Taylor series expansion. For edge $(i+\frac{1}{2}, j)$ this gives

$$U^{n+\frac{1}{2},L}_{i+\frac{1}{2},j} = U^n_{ij} + \frac{\Delta x}{2} U^n_{x,ij} + \frac{\Delta t}{2} U^n_{t,ij}$$

extrapolating from (i, j), and

$$U^{n+\frac{1}{2},R}_{i+\frac{1}{2},j} = U^n_{i+1,j} - \frac{\Delta x}{2} U^n_{x,i+1,j} + \frac{\Delta t}{2} U^n_{t,i+1,j}$$

extrapolating from $(i+1, j)$, with analogous formulae for the other edges. The differential equation (1.1) is then used to eliminate the time derivatives to obtain

$$U^{n+\frac{1}{2},L}_{i+\frac{1}{2},j} = U^n_{ij} + \left(\frac{\Delta x}{2} - \frac{u_{ij}\Delta t}{2}\right) U^n_{x,ij} - \frac{\Delta t}{2}(\widehat{vU_y})_{ij} + \frac{\Delta t}{2}\left(\varepsilon \Delta^h U^n_{ij} - Gp^{n-\frac{1}{2}}_{ij}\right), \tag{3.1}$$

$$\begin{aligned} U^{n+\frac{1}{2},R}_{i+\frac{1}{2},j} &= U^n_{i+1,j} - \left(\frac{\Delta x}{2} + \frac{u_{i+1,j}\Delta t}{2}\right) U^n_{x,i+1,j} \\ &\quad - \frac{\Delta t}{2}(\widehat{vU_y})_{i+1,j} + \frac{\Delta t}{2}\left(\varepsilon \Delta^h U^n_{i+1,j} - Gp^{n-\frac{1}{2}}_{i+1,j}\right). \end{aligned} \tag{3.2}$$

Here, Δ^h is the standard five-point finite difference approximation to the Laplacian. Equations (3.1)-(3.2) represent the final form of the predictor. Analogous formulae are used to predict values at each of the other edges of the cell. In evaluating these terms the first-order derivatives normal to the edge (in this case U_x) are evaluated using a monotonicity-limited fourth-order centered-difference slope approximation [8]. The limiting is done on the components of the velocity individually.

The transverse derivative terms ($\widehat{vU_y}$ in this case) are evaluated by first extrapolating from above and below to construct edge states, using normal derivatives only, and then choosing between these states using the upwinding procedure defined below. In particular, we define

$$\widehat{U}^B_{i,j+\frac{1}{2}} = U^n_{ij} + \left(\frac{\Delta y}{2} - \frac{v_{ij}\Delta t}{2}\right) U_{y,ij},$$

$$\widehat{U}^T_{i,j+\frac{1}{2}} = U^n_{i,j+1} - \left(\frac{\Delta y}{2} + \frac{v_{i,j+1}\Delta t}{2}\right) U_{y,i,j+1},$$

where U_y are limited slopes in the y-direction, with similar formulae for the lower edge of B_{ij}. Using the upwinding procedure we first define the normal advective velocity on the edge:

$$\widehat{v}_{adv} = \begin{cases} \widehat{v}^B & \text{if } \widehat{v}^B > 0, \quad \widehat{v}^B + \widehat{v}^T > 0, \\ 0 & \text{if } \widehat{v}^B \le 0, \quad \widehat{v}^T \ge 0, \\ \widehat{v}^T & \text{otherwise.} \end{cases}$$

(We suppress the $i, j + \frac{1}{2}$ spatial indices on the bottom and top states here and in the next equation.) We now upwind U based on $\widehat{v}_{adv}$:

$$\widehat{U}_{i,j+\frac{1}{2}} = \begin{cases} \widehat{U}^B & \text{if } \widehat{v}_{adv} > 0, \\ \frac{1}{2}(\widehat{U}^B + \widehat{U}^T) & \text{if } \widehat{v}_{adv} = 0, \\ \widehat{U}^T & \text{if } \widehat{v}_{adv} < 0. \end{cases}$$

After constructing $\widehat{U}_{i,j-1/2}$ in a similar manner, we use these upwind values to form an approximation to the transverse derivative in (3.2):

$$\widehat{vU_y} \approx \frac{1}{2}(\widehat{v}_{i,j+\frac{1}{2}} + \widehat{v}_{i,j-\frac{1}{2}})(\widehat{U}_{i,j+\frac{1}{2}} - \widehat{U}_{i,j-\frac{1}{2}}).$$

Corrector. In the corrector we use the identical upwinding procedure to that used for the transverse derivatives to choose the appropriate states $U_{i+1/2,j}$ given the left and right states $U^{n+1/2,L}_{i+1/2,j}$ and $U^{n+1/2,R}_{i+1/2,j}$. We follow a similar procedure to construct $U_{i-1/2,j}$, $U_{i,j+1/2}$, and $U_{i,j-1/2}$. Finally, we use these values to form an approximation of the convective derivatives in (1.1)

$$uU_x + vU_y \approx \frac{1}{2}(u_{i+\frac{1}{2},j} + u_{i-\frac{1}{2},j})(U_{i+\frac{1}{2},j} - U_{i-\frac{1}{2},j}) + \frac{1}{2}(v_{i,j+\frac{1}{2}} + v_{i,j-\frac{1}{2}})(U_{i,j+\frac{1}{2}} - U_{i,j-\frac{1}{2}}).$$

The Godunov method is an explicit difference scheme and, as such, requires a time-step restriction. A linear, constant-coefficient analysis shows that for stability we must require

$$\max_{ij}\left(\frac{|u_{ij}|\Delta t}{\Delta x}, \frac{|v_{ij}|\Delta t}{\Delta y}\right) = \sigma \le 1,$$

where σ is the CFL number. The time-step restriction of the Godunov method is used to set the time step for the overall algorithm.

A weak nonlinear instability for $\sigma > 0.5$ has been observed [2]; it is believed to result from the use of the lagged pressure gradient in the predictor-corrector algorithm as described above. An alternate form of the corrector incorporates a MAC projection as in [2] to ensure that the edge velocities satisfy

$$D^{MAC}U = \frac{u_{i+1/2,j} - u_{i-1/2,j}}{\Delta x} + \frac{v_{i,j+1/2} - v_{i,j-1/2}}{\Delta y} = 0.$$

This projection is applied immediately before the construction of the convective derivatives and eliminates the instability for $.5 < \sigma \le 1$. Convergence results will be presented for calculations with and without the MAC projection.

Parabolic approximation. Once the advection terms are evaluated at every cell B_{ij}, to complete the solution of (2.1) we must solve the diffusion part of the system with the pressure and advective terms treated as source terms. For this step we use a standard five-point discretization of the Laplacian with a modification at the boundary that computes a second-order approximation of Laplacian that reflects the given boundary data. The resulting system for each velocity component is solved using multigrid.

4. Discretization of the projection. In this section we describe the numerical approximation of the projection. The projection is based on a finite element formulation. In particular, we consider the scalar pressure field to be a C^0 function that is a bilinear function over each cell; i.e., the pressure is in

$$S^h = M_0^1(x) \otimes M_0^1(y),$$

where $M_s^t(x)$ is the space of polynomials of degree t in the x-direction on each cell with C^s continuity at x-edges. For the velocity space we define

$$\mathbf{V}^h = \mathbf{V}^{h,x} \times \mathbf{V}^{h,y},$$

where $\mathbf{V}^{h,x} = M_{-1}^0(x) \otimes M_{-1}^1(y)$ and $\mathbf{V}^{h,y} = M_{-1}^1(x) \otimes M_{-1}^0(y)$; i.e., u is piecewise constant in x and a discontinuous linear function of y in each cell, with a similar form for v.

As noted above, our basic approximation represents U in terms of cell averages. The velocity space $\mathbf{V}^h$ contains additional functions that represent the linear variation within each cell. These additional degrees of freedom make $\mathbf{V}^h$ large enough to contain $\nabla\phi$ for $\phi \in S^h$. We establish a correspondence between these two representations by introducing an orthogonal decomposition of $\mathbf{V}^h$. In particular, for each $V \in \mathbf{V}^h$ we define a piecewise constant component

$$\overline{V}_{ij} = \frac{1}{Vol\ B_{ij}} \int_{B_{ij}} V\, d\mathbf{x}$$

and the variation

$$V^{\perp} = V - \overline{V}$$

so that for each cell B_{ij}, $\int_{B_{ij}} V^{\perp} d\mathbf{x} = 0$. By construction these two components are orthogonal in L^2, so they can be used to define a decomposition of $\mathbf{V}^h$ into two components

$$\mathbf{V}^h = \overline{\mathbf{V}}^h \oplus \mathbf{V}^{h\perp}, \tag{4.1}$$

where $\overline{\mathbf{V}}^h$ and $\mathbf{V}^{h\perp}$ represent the cell averages and the orthogonal linear variation, respectively. The decomposition of $\mathbf{V}^h$ induces a decomposition of $\nabla\phi$ for all $\phi \in S^h$; namely,

$$(\nabla\phi)_{ij} = (\overline{\nabla\phi})_{ij} + (\nabla\phi)_{ij}^{\perp},$$

where

$$G\phi_{ij} \equiv (\overline{\nabla\phi})_{ij} = \left(\frac{\phi_{i+1/2,j+1/2} + \phi_{i+1/2,j-1/2} - \phi_{i-1/2,j+1/2} - \phi_{i-1/2,j-1/2}}{2\Delta x}, \right.$$

$$\left. \frac{\phi_{i+1/2,j+1/2} + \phi_{i-1/2,j+1/2} - \phi_{i+1/2,j-1/2} - \phi_{i-1/2,j-1/2}}{2\Delta y} \right),$$

$$(\nabla\phi)_{ij}^{\perp} = (\phi_{i+\frac{1}{2},j+\frac{1}{2}} + \phi_{i-\frac{1}{2},j-\frac{1}{2}} - \phi_{i-\frac{1}{2},j+\frac{1}{2}} - \phi_{i+\frac{1}{2},j-\frac{1}{2}}) \left(\frac{y_{ij}}{\Delta x}, \frac{x_{ij}}{\Delta y} \right),$$

where $\phi_{i+1/2,j+1/2}$ represent the nodal values of ϕ. Here x_{ij} and y_{ij} are local variables, defined on each cell such that $x_{ij} = y_{ij} = 0$ at the center of B_{ij}, $x_{ij} = \pm\frac{1}{2}$ at the left and right edges of B_{ij}, and $y_{ij} = \pm\frac{1}{2}$ at the top and bottom edges of B_{ij}.

We now define a weak form of the projection on $\mathbf{V}^h$, based on a weak divergence on $\mathbf{V}^h$. In particular, we define a vector field V^d in $\mathbf{V}^h$ to be divergence-free if

$$\int_\Omega V^d \cdot \nabla\psi \, d\mathbf{x} = 0 \qquad \forall\psi \in S^h. \tag{4.2}$$

With this definition we can then project any vector field V onto a gradient $\nabla\phi$ and weakly divergence-free field V^d (with vanishing normal velocities on boundaries) by solving

$$\int_\Omega \nabla\phi(\mathbf{x}) \cdot \nabla\psi_{i+\frac{1}{2},j+\frac{1}{2}}(\mathbf{x}) \, d\mathbf{x} = \int_\Omega V \cdot \nabla\psi_{i+\frac{1}{2},j+\frac{1}{2}}(\mathbf{x}) \, d\mathbf{x} \qquad \forall\psi_{i+\frac{1}{2},j+\frac{1}{2}}(\mathbf{x}) \tag{4.3}$$

for

$$\phi(\mathbf{x}) = \sum_{i,j} \phi_{i+\frac{1}{2},j+\frac{1}{2}} \psi_{i+\frac{1}{2},j+\frac{1}{2}}(\mathbf{x})$$

and setting $V^d = V - \nabla\phi$. Here the ψ's are the standard basis functions for S^h; namely, $\psi_{i+1/2,j+1/2}(\mathbf{x})$ is the piecewise bilinear function having node values $\psi_{i+1/2,j+1/2} \cdot (\mathbf{x}_{k+1/2,\ell+1/2}) = \delta_{ik}\delta_{j\ell}$. We note that with a suitable normalization, the right-hand side of (4.3) defines a nodal value of the divergence of V at node $(i + \frac{1}{2}, j + \frac{1}{2})$; in particular, by using (4.3) we are implicitly defining the discrete divergence by

$$(DV)_{i+\frac{1}{2},j+\frac{1}{2}} = -\frac{1}{\overline{\psi}} \int_\Omega V \cdot \nabla\psi_{i+\frac{1}{2},j+\frac{1}{2}}(\mathbf{x}) \, d\mathbf{x}, \tag{4.4}$$

where $\overline{\psi} = \int_\Omega \psi_{i+1/2,j+1/2}(\mathbf{x}) \, d\mathbf{x}$.

When this approximate projection is applied in the context of our fractional step scheme, the vector field V to which the projection is applied is given by

$$V = \overline{V} \equiv \frac{U^* - U^n}{\Delta t}, \quad V^\perp = 0. \tag{4.5}$$

(In the predictor we have only determined the quantity $\overline{U}^*$, so we make the implicit assumption here that $U^{\perp,*} = U^{\perp,n}$.) In this case, since D operates on a "barred" vector, the divergence is simply

$$(D\overline{V})_{i+\frac{1}{2},j+\frac{1}{2}} = \frac{V^x{}_{i+1,j} + V^x{}_{i+1,j+1} - V^x{}_{i,j} - V^x{}_{i,j+1}}{2\Delta x} + \frac{V^y{}_{i,j+1} + V^y{}_{i+1,j+1} - V^y{}_{i,j} - V^y{}_{i+1,j}}{2\Delta y},$$

where V^x and V^y are the x- and y-components of V, respectively. With $\overline{V}$ defined by (4.5), $V^x = \frac{u^* - u^n}{\Delta t}$ and $V^y = \frac{v^* - v^n}{\Delta t}$. We solve (4.3) for ϕ to obtain an update for p; i.e.,

$$\phi(\mathbf{x}) = \delta p^n \equiv p^{n+\frac{1}{2}} - p^{n-\frac{1}{2}}.$$

For the purposes of the fractional-step scheme we then define

$$\overline{V}^{d} = \overline{V} - \overline{G\phi} \tag{4.6}$$

as the approximation of $\frac{\overline{U}^{n+1} - \overline{U}^{n}}{\Delta t}$ in (2.2).

The vector field $\overline{V}^{d}$ is only approximately divergence-free; i.e., $D\overline{V}^{d} \neq 0$ even in the weak sense of (4.2). The quantity that is weakly divergence-free is $V^d = V - \nabla\phi$; however, in general, $DV^{d^{\perp}} \neq 0$; consequently, $D\overline{V}^{d} \neq 0$. Effectively, what we have done in writing (4.6) is perform the vector field decomposition into a divergence-free component and the gradient of a scalar on $\mathbf{V}^h$ and project the weakly divergence-free component onto $\overline{\mathbf{V}}^{h}$ using the characterization of $\mathbf{V}^h$ given by (4.1).

In order to estimate the effect of only approximately enforcing discrete incompressibility we will rewrite the result of the projection as

$$\frac{\overline{U}^{*} - \overline{U}^{n}}{\Delta t} = \frac{U^{n+1} - U^{n}}{\Delta t} + \nabla \delta p^{n}, \tag{4.7}$$

$$DU^{n+1} = 0,$$

where we implicitly assume $DU^n = 0$, although $D\overline{U}^{n} \neq 0$. Since $D\overline{U}^{n} = -DU^{n^{\perp}}$, we can estimate how errors accumulate by examining the behavior of $U^{n^{\perp}}$. To accomplish this we break (4.7) into its component pieces:

$$\frac{\overline{U}^{*} - \overline{U}^{n}}{\Delta t} = \frac{\overline{U}^{n+1} - \overline{U}^{n}}{\Delta t} + \frac{U^{\perp, n+1} - U^{\perp, n}}{\Delta t} + \nabla(p^{n+\frac{1}{2}} - p^{n-\frac{1}{2}}).$$

If we then use the characterization of $\mathbf{V}^h$ given by (4.1), we obtain

$$\overline{U}^{n+1} = \overline{U}^{n} + \Delta t \left(\frac{\overline{U}^{*} - \overline{U}^{n}}{\Delta t} + Gp^{n-\frac{1}{2}} - Gp^{n+\frac{1}{2}} \right)$$

and

$$U^{n+1^{\perp}} = U^{n^{\perp}} + \Delta t ((\nabla p^{n-\frac{1}{2}})^{\perp} - (\nabla p^{n+\frac{1}{2}})^{\perp}) \approx U^{n^{\perp}} - \Delta t^2 (\nabla p_t^n)^{\perp}.$$

This evolution equation for $U^{\perp}$ places limits on the growth of $U^{\perp}$. The quantity $(\nabla p)^{\perp}$ is uniquely defined by the node values $p_{i+1/2, j+1/2}$. As long as the pressure remains sufficiently smooth in space and time, $U^{\perp}$ is well behaved as well and scales with $\Delta t \; h = \sigma h^2$. The factor of Δt comes from the evolution equation for $U^{\perp}$; the factor of h comes from the fact that for smooth ψ, $|(\nabla\psi)^{\perp}| \leq O(h)$. Thus $DU^n = 0$ guarantees that $D\overline{U}^{n} = O(h^2)$ as long as the CFL condition is enforced. We monitored $D\overline{U}$ for the convergence study presented in the next section and verified the second-order accuracy computationally.

The linear system associated with the solution of (4.3) is the standard bilinear finite element stiffness matrix for Poisson's equations. We solve this system using standard multigrid methods (see [5]); in particular, we use the standard V-cycle with Jacobi relaxation. This procedure reduces the residuals by approximately a factor of five per V-cycle.

The left-hand side of (4.3) is, in discrete form, a nine-point stencil approximating the Laplacian of ϕ. We note that it is possible, using a similar development, to construct an approximate projection in which the left-hand side of (4.3) is the standard five-point finite difference Laplacian. This construction is based on approximating pressure as a piecewise linear function on triangles. We define a triangulation of the domain by connecting lower-left corners of each grid cell to upper-right corners. With this definition we define $\mathbf{V}^h$ to

TABLE 1
Convergence rates—MAC-less predictor.

Case	16-32	Rate	32-64	Rate	64-128	Rate	128-256
Euler	1.44e-2	2.22	3.08e-3	2.36	5.98e-4	2.17	1.33e-4
Re 100	6.70e-3	2.06	1.61e-3	1.94	4.20e-4	1.90	1.12-4
Stokes	4.95e-3	1.99	1.25e-3	1.99	3.14e-4	1.91	8.39-5

TABLE 2
Convergence rates—predictor using MAC.

Case	16-32	Rate	32-64	Rate	64-128	Rate	128-256
Euler	1.44e-2	2.26	3.00e-3	2.35	5.90e-4	2.18	1.30e-4
Re 100	6.10e-3	2.15	1.37e-3	1.91	3.63e-4	1.87	9.93e-5

be piecewise constant functions on each triangle. As in the bilinear case, we establish the relationship between the projection and the upwind procedure by defining a decomposition of $\mathbf{V}^h$ similar to (4.1); namely,

$$\mathbf{V}^h = \overline{\mathbf{V}}^h \oplus \mathbf{V}^{h^\perp},$$

where $\overline{\mathbf{V}}^h$ is defined as before and $\mathbf{V}^{h^\perp}$ is piecewise constant on each triangle and averages to zero over each of the original rectangular grid cells. This alternate form also leads to an approximate projection in which the vector field decomposition is performed in $\mathbf{V}^h$ and the result is projected onto $\overline{\mathbf{V}}^h$; however, in this case (4.3) for ϕ gives

$$\frac{\phi_{i+3/2,j+1/2} + \phi_{i-1/2,j+1/2} + \phi_{i+1/2,j+3/2} + \phi_{i+1/2,j-1/2} - 4\phi_{i+1/2,j+1/2}}{h^2} = (DV)_{i+\frac{1}{2},j+\frac{1}{2}},$$

where $DV_{i+1/2,j+1/2}$ is defined as in (4.4). We have verified the second-order accuracy of the method with this five-point stencil and observed that the L_2-errors between calculations with different resolution using the five-point stencil differ by less than 2% from the errors shown in Tables 1 and 2.

Boundary conditions for the projection. For homogeneous boundary conditions we want $V^d \cdot \mathbf{n} = 0$ on $\partial\Omega$. This condition is enforced (weakly) by including ψ's that are nonzero on the boundary of Ω in S^h, which imposes a natural treatment of the boundary conditions for the Poisson equation. If we formally integrate (4.3) by parts we see that ϕ satisfies

$$\Delta\phi = DV \quad \text{on } \Omega,$$

$$\frac{\partial\phi}{\partial n} = V \cdot \mathbf{n} \quad \text{on } \partial\Omega.$$

For inflow and outflow, we want to specify $V^d \cdot \mathbf{n}$ on part of the boundary (inflow), which we will refer to as Γ_1, and have outflow on Γ_2. On the outflow face we want to impose the condition that there are no net forces accelerating the fluid parallel to the outflow face. In this simple setting we can accomplish this by setting $\phi = 0$ on Γ_2. Thus, for outflow we restrict S^h to include only those functions that vanish at points on Γ_2.

We do allow ψ's to have support on Γ_1, so we solve for ϕ's on Γ_1, as in the homogeneous boundary condition case; however, we must augment (4.3) with boundary terms to reflect the specified boundary data. In particular, we add a boundary term to (4.3) to obtain

$$\int_\Omega V \cdot \nabla\psi \, d\mathbf{x} = \int_\Omega \nabla\phi \cdot \nabla\psi \, d\mathbf{x} + \int_{\Gamma_1} \psi V^d \cdot \mathbf{n} \, ds.$$

We substitute the boundary conditions for V^d on Γ_1, subtract this quantity from the left-hand side, and proceed with the projection. As can be seen by formally integrating by parts, we now have

$$\frac{\partial\phi}{\partial n} = V \cdot \mathbf{n} - V^d \cdot \mathbf{n} \quad \text{on } \Gamma_1,$$

which enforces the desired condition. Note that the boundary condition for V^d is $V^d \cdot \mathbf{n} = (u_{in}^{n+1} - u_{in}^n)/\Delta t$, where u_{in} is the time-dependent inflow velocity.

5. Numerical results. In this section we first present numerical results to demonstrate the second-order convergence of the method presented here. We also apply the method to computation of a forced, unstable shear layer to illustrate the capabilities of the method in a more realistic setting.

For the convergence study we use the same initial data as was used in [1]. We initialize the data with a smooth velocity field

$$u(x, y) = -\sin^2(\pi x)\sin(2\pi y),$$

$$v(x, y) = \sin^2(\pi y)\sin(2\pi x),$$

in the unit square. The velocity satisfies homogeneous Dirichlet boundary conditions.

Calculations were run to $t = 0.5$ with $\Delta x = \Delta y = 1/2^n$ for $n = 4, 5, 6, 7, 8$. The time step was restricted by the stability criterion to be $\Delta t = .5\Delta x$, which approximates CFL $= 0.5$. To estimate the convergence rate, we find the L_2-norm of the difference between the velocity obtained on each grid and the velocity obtained on the next finer grid; then take $\log_2$ of the ratio of these norms. The convergence rates for velocity are given in Table 1 for the MAC-less predictor, and in Table 2 for the predictor that uses the MAC projection. (We also present, in Table 1, results for Stokes flow.)

The second set of calculations is a direct comparison with the experimental study of Ho and Huang [12] on the nonlinear instability of a forced shear layer. The forcing is introduced using an upstream perturbation based on the stability analysis of Monkewitz and Huerre [17]. We note that a number of other authors have also treated this problem; see, for example, Kuhl et al. [14], who include a survey of the other literature.

Our computations were performed on a 512 × 128 uniform grid, with $\varepsilon = 0$. In Figures 1 and 2 we present contours of vorticity for two cases corresponding to the fundamental perturbation frequency (ω_1) plus a subharmonic. The cases correspond to the first (ω_2) and second (ω_3) subharmonics, respectively. The largest perturbation amplitude was 1% for the fundamental with reduced amplitudes for the subharmonics as given in [17]. The numerical results show the instability of the shear layer and the downstream vortex merging pattern. The first case consists of a simple pairing; in the second case the vortices merge in groups of three with the latter two vortices merging first. The prediction of the correct merging patterns and the near periodicity of the results serve to validate the numerical method on a realistic flow problem.

We also compute the flow perturbed by the fundamental plus nine subharmonics. Computational results for this case are shown in Figure 3. In this case we see a substantially more complex pattern of merger, analogous to the broadly forced instability of this shear layer

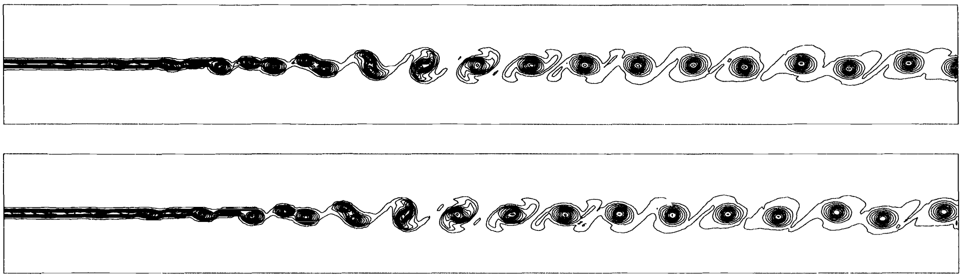


FIG. 1. $\omega_1 + \omega_2$ *forcing at times* T = 1333, 1369.

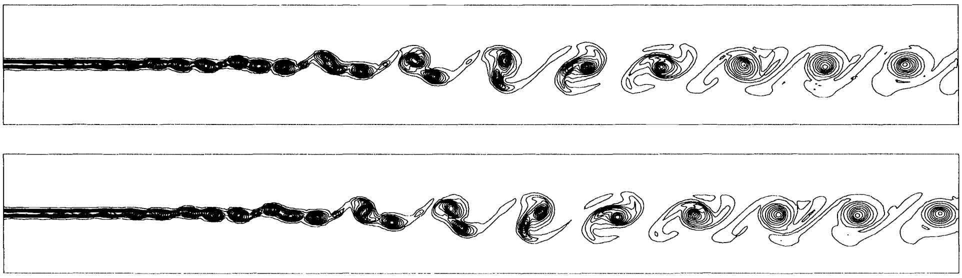


FIG. 2. $\omega_1 + \omega_3$ *forcing at times* T = 1319, 1354.

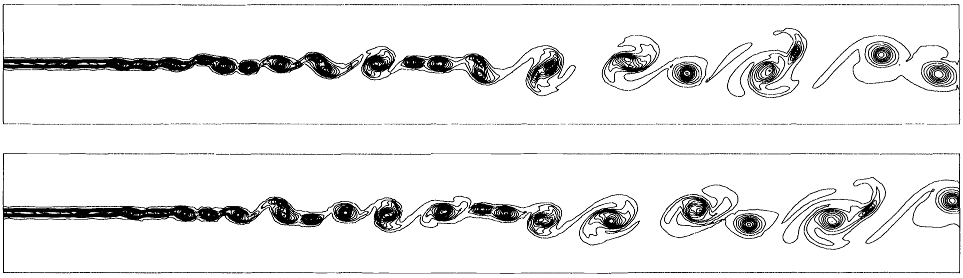


FIG. 3. *Ten frequency forcing forcing at times* T = 1281, 1351.

investigated by Oster and Wygnanski [18]. Although detailed comparisons for this case are beyond the scope of this paper, we note that the spreading rate in the computation agrees well with the experimental value of 11 degrees. Also note that we see no difficulties at inflow and outflow boundaries for either this case or the preceding cases.

6. Conclusions. Traditionally, projection methods for discretizing the time-dependent incompressible Navier-Stokes equations have exactly applied a discrete form of the projection; i.e., the velocity field returned from the projection satisfies a discrete constraint $DU = 0$ for some discrete divergence operator D. In this paper we have shown that this requirement can be relaxed. We have defined a projection based on a finite element construction that only enforces the divergence-free condition to second-order accuracy. The overall accuracy of the method is essentially no different than previous versions of the methodology that exactly imposed a discrete form of the constraint. By relaxing this requirement we have been able to define a projection for which the linear system that must be solved corresponds to a conventional discretization of the Laplacian but for which the velocities are cell-centered, enabling the use of modern upwind techniques for treating the advective derivatives.

This simplified projection is easily generalized to three dimensions and to variable density flows (see [3]). This generalization will be discussed in a forthcoming paper. We are also using this approach to develop an adaptive mesh version of the projection method. In this context we can exploit the similarity with standard finite element techniques in order to use iterative methods developed in the elliptic community for adaptivity.

REFERENCES

[1] J. B. Bell, P. Colella, and H. M. Glaz, *A second-order projection method for viscous, incompressible flow*, in Proc. 8th AIAA Computational Fluid Dynamics Conference, Honolulu, HI, June 9-11, 1987.

[2] J. B. Bell, P. Colella, and L. H. Howell, *An efficient second-order projection method for viscous incompressible flow*, in Proc. 10th AIAA Computational Fluid Dynamics Conference, Honolulu, HI, June 24-27, 1991.

[3] J. B. Bell and D. L. Marcus, *A second-order projection method for variable-density flows*, J. Comput. Phys., 101 (1992), pp. 334-348.

[4] J. B. Bell, J. M. Solomon, and W. G. Szymczak, *A second-order projection method for the incompressible Navier–Stokes equations on quadrilateral grids*, in Proc. 9th AIAA Computational Fluids Dynamics Conference, Buffalo, NY, June 14-16, 1989.

[5] W. L. Briggs, *A Multigrid Tutorial*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1987.

[6] A. J. Chorin, *Numerical solution of the Navier–Stokes equations*, J. Math. Comput., 22 (1968), pp. 745-762.

[7] P. Colella, *Private communcation*, 1992.

[8] ———, *A direct Eulerian MUSCL scheme for gas dynamics*, SIAM J. Comput., 6 (1985), pp. 104-117.

[9] ———, *A multidimensional second order Godunov scheme for conservation laws*, J. Comput. Phys., 87 (1990), pp. 171-200.

[10] M. Fortin, *Numerical solutions of the steady state Navier–Stokes equations*, in Numerical Methods in Fluid Dynamics, AGARD-LS-48, North Atlantic Treaty Org., Advisory Group for Aerospace Research & Development, Lecture Series 48, J. J. Smolderen, ed., Neuilly-sur-Seine, France, 1972.

[11] F. H. Harlow and J. E. Welch, *Numerical calculation of time-dependent viscous incompressible flow of fluids with free surfaces*, Phys. Fluids, 8 (1965), pp. 2182-2189.

[12] C. M. Ho and L. S. Huang, *Subharmonics and vortex merging in mixing layers*, J. Fluid Mech., 119 (1982), pp. 443-473.

[13] L. H. Howell and J. B. Bell, *An adaptive-mesh projection method for viscous incompressible flow*, SIAM J. Sci. Comput., to appear.

[14] A. L. Kuhl, K.-Y. Chien, R. E. Ferguson, H. M. Glaz, and P. Colella, *Inviscid Dynamics of Unstable Shear Layers*, Report RDA-TR-161604-004, R&D Associates, Marina del Rey, CA, April 1988.

[15] M. Lai, J. B. Bell, and P. Colella, *A projection method for combustion in the zero mach number limit*, in Proc. 11th AIAA Computational Fluid Dynamics Conference, Orlando, FL, July 6-9, 1993, pp. 776-783.

[16] M. F. Lai, *A Projection Method for Reacting Flow in the Zero Mach Number Limit*, Ph.D. thesis, University of California, Berkeley, CA, 1993.

[17] P. A. Monkewitz and P. Huerre, *Influence of the velocity ratio on the spatial instability of mixing layers*, Phys. Fluids, 25 (1982), pp. 1137-1143.

[18] D. Oster and I. Wygnanski, *The forced mixing layer between parallel streams*, J. Fluid Mech., 123 (1982), pp. 91-130.

[19] R. B. Pember, A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, and M. Lai, *A higher-order projection method for the simulation of unsteady turbulent nonpremixed combustion in an industrial burner*, in Proc. 8th International Symposium on Transport Phenomena in Combustion, San Francisco, CA, July 16-20, 1995.

[20] J. Strikwerda, *Finite difference methods for the Stokes and Navier–Stokes equations*, SIAM J. Sci. Statist. Comput., 5 (1984), pp. 56-67.

SIAM J. SCI. COMPUT.
Vol. 17, No. 2, pp. 370-397, March 1996

RESURRECTING CORE SPREADING VORTEX METHODS: A NEW SCHEME THAT IS BOTH DETERMINISTIC AND CONVERGENT*

LOUIS F. ROSSI†

Abstract. A basic core spreading vortex scheme is inconsistent but can be corrected with a splitting algorithm, yielding a deterministic and efficient grid-free method for viscous flows. The splitting algorithm controls the consistency error by maintaining small vortex core sizes. Routine analysis will show that the core spreading method coupled to this splitting process is convergent in L^p spaces. Analysis of the linearized residual operator establishes the uniform convergence of this method when the exact flow field is known. A sequence of examples demonstrates the sensitivity of the method to numerical parameters as the computed solution converges to the exact solution. These experimental results agree with the linear convergence theory. Finally, direct comparisons between the traditional random walk vortex method and the new method indicate that the new method has several advantages while requiring the same computational effort.

Key words. vortex methods, vorticity dynamics, computational fluid dynamics, convergence theory

AMS subject classifications. 35Q30, 41A25, 41A30, 65D99, 65M12, 65M50, 65M60, 76D05

1. Introduction.

Core spreading algorithms have not been studied since 1985 when C. Greengard proved that the basic core spreading method would not converge [8]. There are many convergent methods for incorporating viscosity into vortex simulations, but none of them are as simple as a core spreading method. A core spreading algorithm is fully deterministic, allowing precise error control and faster convergence than stochastic methods. Also, these algorithms do not rely on operator splitting, the division of the Navier-Stokes equations into separate convective and diffusive processes. Furthermore, a core spreading method is fully localized and grid free, permitting fast parallel execution, free from flow geometry considerations. Neither the circulation redistribution schemes of Mas and Gallic nor the random walk methods of Chorin have both of these advantages [3], [14]. In this paper, we introduce a convergent core spreading vortex method that is both deterministic and localized.

While this new method is effective for unbounded flows, it was developed specifically for flows around general boundaries [15]. One weakness of the split-step techniques more commonly used for vortex methods is that they do not satisfy both normal and tangential boundary conditions at any given moment in the simulations except under very special conditions [3], [4], [5], [19]. Rather, they leapfrog from satisfying the normal flow condition to locally satisfying the tangential flow condition. Sethian has compiled a comprehensive bibliography on vortex methods covering these and other issues [18]. The new method is not a split-step method so it is possible to satisfy both boundary conditions using ideas similar to those of Chorin and Sethian though a thorough discussion of these methods is beyond the scope of this paper. This paper examines only the properties of this method for solving unbounded flows with the understanding that these may well be flows in the interior of some bounded region.

The basic core spreading method is corrected by introducing a process called "adaptive spatial refinement." This new vortex method is called the corrected core spreading vortex method (CCSVM). Since the width of each basis function, or blob, is the critical convergence parameter for a vortex method, it is necessary that blobs not grow too wide. Adaptive spatial refinement approximates a single vortex element or blob with several thinner blobs. While Lu and Ross investigated the possibility of using vortex splitting to approximate diffusive processes [13], the instantaneous reconfiguration of vortex elements has not been considered

*Received by the editors August 26, 1993; accepted for publication (in revised form) October 18, 1994. This work was supported by U.S. Air Force Office of Scientific Research ASSERT grant 496092J0324DEF.

†Department of Engineering Sciences and Applied Mathematics, Northwestern University, 2145 Sheridan Road, Evanston, IL 60208-3125 (LF-Rossi@nwu.edu).

as a method for reducing the core width. Surprisingly, it is possible to approximate any vortex element to any specified tolerance with a formation of arbitrarily thinner blobs. Thus, refinement is a technique for replacing one vortex element with many but has nothing to do with the physical process of a single region of vorticity splitting into several distinct regions. This technique has many similarities to the local regridding concepts introduced by Hou, Lowengrub, and Shelley, but core spreading with adaptive spatial refinement does not require the use of an underlying Eulerian coordinate system [11]. This paper establishes the linear convergence of CCSVM to the Navier-Stokes equations.

The work of Beale and Majda already establishes the full nonlinear convergence of the new method in the weaker L^p norms [2], but this paper will focus on the L^∞ (uniform) convergence of the CCSVM in the linear case. Goodman and Hou established uniform convergence of 2D vortex methods to Euler equations, but no such results exist for the Navier-Stokes equations partially due to the stochastic convergence properties of the more popular random walk method of modeling diffusion [7]. Uniform convergence is a more desirable result because it represents the maximum deviation from the exact solution anywhere in the domain. Also, the standard vortex method convergence theory expresses convergence in terms of the velocity field. By using a nonstandard approach, convergence can express itself as the difference between computed and exact vorticity fields. This result is developed by decoupling the velocity field from the vorticity field. The uniform convergence theory highlights two convergence parameters. The first parameter, l, is the maximum core width in the simulation. Hald used this parameter in his first inviscid convergence proof [9] and has continually appeared in subsequent vortex method analyses. If a vortex element or blob grows larger than l, the refinement scheme will replace it with several thinner elements. The other numerical parameter, α, controls the accuracy and frequency of the refinement process and is unique to CCSVM, though some associations can be drawn to parameters in other diffusive methods. More accurate and frequent refinement increases the total number of vortex elements in the simulation, as would be expected.

To control the problem size, many computational elements can be combined into a single one under certain circumstances in a process called "vortex fusion." It is clear that two blobs sharing the same location and core width could be replaced by a single blob with no change at all to the induced vorticity field. An efficient code would merge these two blobs without any loss of accuracy. Similarly, it can be shown that nearby elements with similar widths can be fused with small errors in the vorticity field. Thus, it is possible to reduce the problem size with precise error control. Since a thorough error analysis of this process is beyond the scope of this paper, only the fundamental results together with a demonstration will be presented in this paper. The full analysis, an explicit fusion algorithm, and detailed examples have been submitted for future publication [16].

In the last section, some examples will highlight the interplay between l and α as well as demonstrate the efficiency and accuracy of this new vortex method. To our knowledge, this is the first time precise measurements of vorticity field convergence have been measured in the uniform norm. These measurements show strong agreement with the linear convergence theory. Also, it will be shown that even a very conservative fusion scheme can increase the efficiency of the method considerably while inflicting a very small loss of accuracy. The ultimate implications are that this method is a fast, accurate and naturally adaptive method for viscous flow computations.

2. Formulation and algorithm.

2. Formulation and algorithm. This method is designed to simulate 2-D, unbounded, incompressible flows. Vortex methods attempt to approximate the vorticity dynamics equation

$$\frac{D\omega}{Dt} = \nu \nabla^2 \omega, \tag{1}$$

where $\frac{D}{Dt} = \partial_t + (\vec{u} \cdot \nabla)$. The velocity field $\vec{u}$ can be determined from the vorticity field using a Green's function. Essentially, one must solve the heat equation in a frame moving with the fluid. A vortex method accomplishes this by representing the full vorticity field as a sum of basis functions.

Unlike most vortex methods, CCSVM does not split the vorticity dynamics equation. Operator splitting breaks equation (1) into

$$\frac{D\omega}{Dt} = 0, \tag{2}$$

$$\partial_t \omega = \nu \nabla^2 \omega. \tag{3}$$

The first equation, the Euler equation, is approximated by a simple vortex method. Chorin handled the second equation by having each vortex undergo a random walk. There are many other ways of approximating diffusive processes without grids, but all involve operator splitting. Even though vortex methods converge to exact solutions of the Navier-Stokes equations when the operator is split, the numerical error will also include a splitting error not present in CCSVM. Furthermore, it is difficult to define concepts such as consistency and stability without keeping the operator intact. Hald notes in his original convergence proof that any distinction between the two properties is artificial [9].

To facilitate a future convergence theory, it is desirable to invent a method which is consistent with equation (1) without relying on operator splitting arguments. For solving the heat equation, Gaussian basis functions are advantageous:

$$\omega(\vec{x}) = \sum_{i=1}^{N} \frac{\gamma_i}{4\pi\sigma_i^2} \exp\left(-\frac{|\vec{x} - \vec{x}_i|^2}{4\sigma_i^2}\right), \tag{4}$$

where γ_i is the circulation of blob i and σ_i is the blob width. Though higher spatial accuracy is achieved with the kernels of Hald, Beale, and Majda, Gaussians maintain self-similarity while approximating the diffusive term in equation (1) [2], [10]. To capture equation (1), one must allow each vortex blob to move and diffuse with the flow, giving rise to the following dynamical system:

$$\dot{\vec{x}}_i = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \sum_{j=1}^{N} \frac{\gamma_i}{2\pi} \frac{\vec{x}_i - \vec{x}_j}{|\vec{x}_i - \vec{x}_j|^2} \left[1 - \exp\left(-\frac{|\vec{x}_i - \vec{x}_j|^2}{4\sigma_i^2}\right)\right], \tag{5}$$

$$\dot{\sigma_i^2} = \nu. \tag{6}$$

The first system, equation (5), arises from Stokes' theorem applied to equation (4). The whole blob will move with the velocity of the blob center. Later, it will be clear that this is the source of all consistency error. The second equation arises from solving the heat equation in a frame moving with the flow.

This system of ODEs is not new in the world of vortex methods. In 1985, Greengard demonstrated that it was inconsistent with the Navier-Stokes equations [8]. The reasons are more or less evident from a simple heuristic argument. The convergence parameter for a vortex method is the core size of the vortex elements. According to equation (6), the cores have sizes of at least $\sqrt{\nu T}$ at the end of the simulation. Hence, it is impossible to force the method to converge except under certain special circumstances discussed further in §3.1.

2.1. Reducing consistency error with adaptive spatial refinement. In order to maintain small blob sizes throughout the simulation, the basic core spreading scheme is coupled to adaptive spatial refinement. This refinement process splits any blob wider that l into a configuration of thinner blobs of width strictly less than l. With a practical and effective refinement process, the general convergence theory of Hald, Beale, Majda, and others asserts that CCSVM will have L^2 convergence of order l^2 when using Gaussian basis functions [2], [9].

However, the uniform norm is desired for many physical applications. Since both the calculated and the exact vorticity field decay exponentially, a uniform bound can be converted to an L^2 bound while the opposite is not true. Finally, a uniform stability result is necessary for the forthcoming uniform convergence theory. It will be shown that refinement induces uniformly bounded disturbances to the vorticity field as numerical parameters grow small.

The numerical parameter $\alpha \in [0, 1]$ controls the accuracy and stability of adaptive spatial refinement. The refinement process will approximate a single vortex element having width l with several blobs of width αl. Therefore, a refinement algorithm takes all vortex elements with core sizes greater than or equal to l and splits them into many, each of which is scaled by a factor of α.

The particular refinement process is somewhat arbitrary. Here, only 1-4 refinement, where one vortex blob splits into four, will be analyzed, but this program of analysis can be carried out for any "refined" configuration. The general strategy is to conserve as many moments as needed to constrain the free variables. To begin, a single vortex element with circulation γ and variance σ^2 will split into four identical vortices, each with a variance of $\alpha^2\sigma^2$. Without loss of generality, one can assume the original vortex is located at the origin. Conservation of the zeroth moment indicates the obvious: the new vortices must have the same circulation as the original. Conservation of the first moment along with the rotational symmetry of the system requires that each vortex have circulation $\frac{\gamma}{4}$ and be centered uniformly along a circle of radius r. Conservation of the second moment constrains the last variable

$$r = 2\sigma\sqrt{1-\alpha^2}. \tag{7}$$

Now, all of the free parameters of 1-4 refinement have been determined, and it is possible to explicitly describe CCSVM.

1. Approximate the exact initial vorticity distribution with vortex elements.
2. Numerically solve equations (5) and (6) to convect and spread each vortex element.
3. Split any element with $\sigma_i > l$ according to the 1-4 refinement process listed above with the refinement radius described in equation (7).
4. Repeat (2) and (3) until reaching time T.

As noted earlier, under the assumption that the field error induced by the refinement process is stable for $0 \leq t \leq T$, the previous work of other investigators such as Beale and Majda guarantees weak convergence of this algorithm. Also, one can see that this method is very similar to the more traditional random walk vortex method (RWVM). On the surface, it would appear that CCSVM and RWVM would require a similar number of numerical calculations given the same N. Experiments in §5 confirm that this is the case.

2.2. The stability of adaptive spatial refinement. While adaptive spatial refinement controls the core size of all elements in the simulation, it also induces small errors into the simulation because there will be a difference between the original vortex element and the new refined configuration. Fortunately, this error is a function of α and will converge to zero as α approaches unity. It is possible to make strict uniform estimates of this refinement-induced field error.

This estimate begins with a function expressing the pointwise difference between the unrefined and refined fields.

$$e(\vec{x}) = \frac{\gamma}{4\pi\sigma^2}\exp\left(-\frac{|\vec{x}|^2}{4\sigma^2}\right) - \sum_{i=1}^{4}\frac{\gamma}{16\pi\alpha^2\sigma^2}\exp\left(-\frac{|\vec{x}-\vec{x}_i|^2}{4\alpha^2\sigma^2}\right), \tag{8}$$

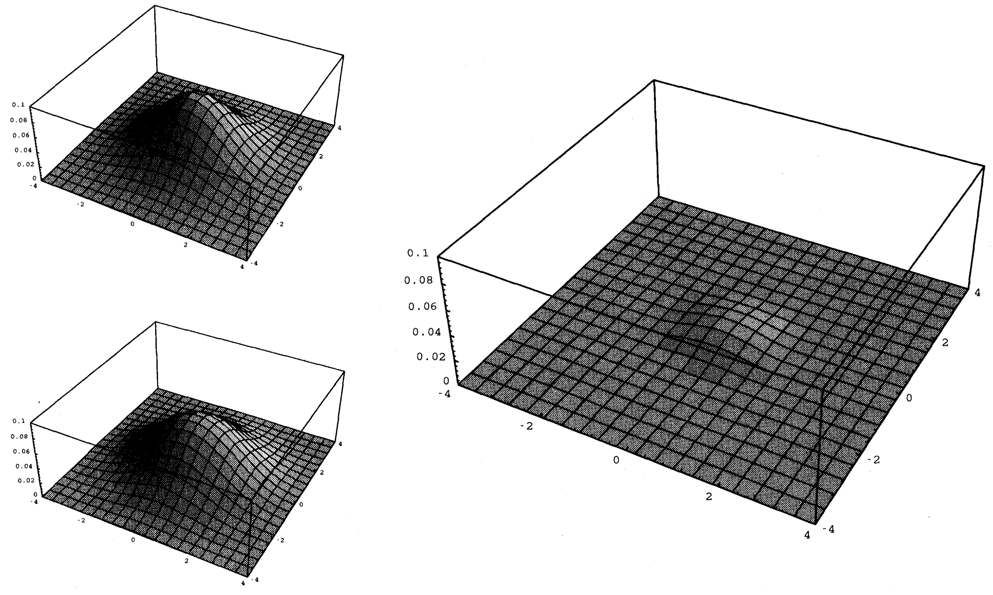


FIG. 1. *Adaptive spatial refinement with $\alpha = 0.8$. Upper left: Gaussian vortex before refinement. Lower left: Gaussian vortex after refinement. Right: difference between unrefined and refined vortices.*

where $\vec{x}_1 = \begin{bmatrix} r \\ 0 \end{bmatrix}$, $\vec{x}_2 = \begin{bmatrix} -r \\ 0 \end{bmatrix}$, $\vec{x}_3 = \begin{bmatrix} 0 \\ r \end{bmatrix}$, and $\vec{x}_4 = \begin{bmatrix} 0 \\ -r \end{bmatrix}$, and r comes from equation (7). Since the goal is to find a uniform norm, a scale transformation can be applied to all the spatial variables in this function as follows:

$$\frac{\vec{x}}{2\sigma} \rightarrow \hat{\vec{x}}. \tag{9}$$

Thus,

$$e(\hat{\vec{x}}) = \frac{\gamma}{4\pi\sigma^2}\left[\exp(-|\hat{\vec{x}}|^2) - \frac{1}{4\alpha^2}\sum_{i=1}^{4}\exp\left(-\frac{|\hat{\vec{x}} - \hat{\vec{x}}_i|^2}{\alpha^2}\right)\right]. \tag{10}$$

Detailed analysis of the term in [ ]'s will show that its maximum value is at the origin for realistically large ($\alpha > 0.43\ldots$) values of α.

Evaluating equation (10) at the origin, the uniform refinement error is

$$\|e(\hat{\vec{x}})\|_\infty = \frac{|\gamma|}{4\pi\sigma^2}\left[1 - \frac{1}{\alpha^2}\exp\left(\frac{\alpha^2 - 1}{\alpha^2}\right)\right]. \tag{11}$$

For any consistent method of initially assigning σ and γ to a vortex element, γ/σ^2 is bounded as the problem size grows. It is appropriate that this estimate only depends on the convergence parameter α and not l because l is strictly a measurement of blob width and does not control the refinement process. It is only when the total evolution of a system of vortex elements is analyzed that a relationship between l and α will emerge.

EXAMPLE 1 ($\alpha = 0.8$). *A vortex element with $\gamma = 1$ and $\sigma = 1$ undergoes refinement. Here, $\|e(\hat{\vec{x}})\|_\infty = 0.0087$, which is better than 40% improvement over $\alpha = 0.7$. The original vortex, the refined vortices, and the error between the two fields are shown in Fig.* 1.

This uniform bound applies to a single refinement event, but a vortex element may undergo many refinement events in its lifetime. The number of refinement events it will undergo is

1

		1		
	2		2	
1		4		1
	2		2	
		1		

				1				
			4		4			
		6		16		6		
	4		24		24		4	
1		16		36		16		1
	4		24		24		4	
		6		16		6		
			4		4			
				1				

FIG. 2. *A system of vortex elements undergoes adaptive spatial refinement n times in a uniform flow. Initially, there is only one vortex element located at the center. The distribution of 4^n vortex elements for $n = 0$ (left), $n = 2$ (middle), and $n = 4$ (right) obeys a double binomial distribution. The grid width is r.*

a function of l, α, the viscosity ν, and the total simulation time T. The time required for a vortex element to grow from a size $\sigma = \alpha l$ to size $\sigma = l$ shall be called "the viscous time step" and is denoted by Δt. After an element reaches a width of l, refinement occurs, and it is again reduced to size αl. Presumably, vortex elements shall go through $\frac{T}{\Delta t}$ of these cycles during a simulation. The viscous time step is determined from equation (6) to be

$$\Delta t = \frac{l^2(1-\alpha^2)}{\nu}. \tag{12}$$

Thus, the total number of refinement events, n, in a simulation of duration T is

$$n = \left[\frac{\nu T}{l^2(1-\alpha^2)}\right]. \tag{13}$$

Since refinement events are coupled to core spreading through viscosity, but not by the flow conditions, it is sufficient to analyze the stability of refinement when $\vec{u}$ is constant.

A stable refinement process is one that induces a controllably small error as the number of refinement events grows large. Therefore, one must consider the history of a single vortex element centered at the origin as spreads and splits many times. This vortex will generate a grid with spacing r from equation (7), but like a checkerboard, only even (odd) spaces will have vortex elements for n even (odd). Without loss of generality, one can assume that n is an even number. It follows from a block-walking argument that the distribution of 4^n children of the original vortex obeys a double binomial distribution [20]. More specifically, the number of vortex elements at position $x = \sqrt{2}r(i - j)$ and $y = \sqrt{2}r(i + j)$ is $\left(ni + \frac{n}{2}\right)\left(nj + \frac{n}{2}\right)$. For examples, see Fig. 2.

Thus, after n refinement events, the original element can be expressed as an array of elements with circulations

$$\gamma(x, y) = \gamma_0 \frac{1}{4^n}\left(ni + \frac{n}{2}\right)\left(nj + \frac{n}{2}\right). \tag{14}$$

This expression can also be interpreted as the probability density function (PDF) of two independent Bernoulli trials with $p = \frac{1}{2}$. In fact, this particular PDF corresponds to RWVM with step size $r = \sqrt{2\nu\Delta t}$ where Δt comes from equation (12). Thus, RWVM with Gaussian basis functions and CCSVM are equivalent when there are many computational elements. The crucial difference lies in the fact that CCSVM is deterministic, and one would only expect to observe differences in performance when the problem size is moderate or small.

The DeMoivre–Laplace theorem relates equation (14) to an exponential distribution for large n [6]:

$$\gamma(x, y) = \frac{\gamma_0}{2\pi\left(\frac{n}{4}\right)} \exp\left[-\frac{i^2 + j^2}{2\left(\frac{n}{4}\right)}\right] + O\left(\frac{1}{n}\right). \tag{15}$$

This approximation converges rather rapidly even for small values of n. Using equation (13) and substituting x and y into the left-hand side of equation (15), one obtains

$$\begin{aligned} \gamma(x, y) &= \gamma_0 \frac{8l^2(1-\alpha^2)}{4\pi\nu T} \exp\left[-\frac{x^2+y^2}{4\nu T}\right] + O[l^2(1-\alpha^2)] \\ &= (\sqrt{2}r)^2 \frac{\gamma_0}{4\pi\nu T} \exp\left[-\frac{x^2+y^2}{4\nu T}\right] + O[l^2(1-\alpha^2)] \end{aligned} \tag{16}$$

for discrete values of x and y. The total field is

$$\omega(\vec{x}) = \sum_{i,j} \gamma(\vec{r}_{i,j}) \frac{1}{4\pi\sigma^2} \exp\left(-\frac{|\vec{x} - \vec{r}_{i,j}|}{4\sigma^2}\right) \tag{17}$$

for $\sigma \in [\alpha l, l]$ and $\vec{r}_{i,j} = \sqrt{2}r\left[\begin{smallmatrix} i-j \\ i+j \end{smallmatrix}\right]$.

Adaptive spatial refinement is stable if this quantity is close to the exact solution for all time in the limit as $\alpha \to 0$ and $\alpha \to 1$. Each $\vec{r}_{i,j}$ can be said to lie in the center of a box $B_{i,j}$ of width $\sqrt{2}r$. These $B_{i,j}$ are the components of a grid (oriented 45^o to the mesh depicted in Fig. 2) naturally generated by refinement. Using this grid, one finds that equation (17) is a Riemann sum of a simpler expression,

$$\begin{aligned} &\omega(\vec{x}) - \frac{\gamma_0}{4\pi(\nu T + \sigma^2)} \exp\left[-\frac{|\vec{x}|^2}{4\pi(\nu T + \sigma^2)}\right] \\ &= \sum_{i,j} \gamma(\vec{r}_{i,j}) \frac{1}{4\pi\sigma^2} \exp\left(-\frac{|\vec{x} - \vec{r}_{i,j}|^2}{4\sigma^2}\right) \\ &\quad - \iint \frac{\gamma_0}{4\pi\nu T} \exp\left(-\frac{|\vec{s}|}{4\nu T}\right) \frac{1}{4\pi\sigma^2} \exp\left(-\frac{|\vec{x} - \vec{s}|}{4\sigma^2}\right) d\vec{s} \\ &= \frac{\gamma_0}{4\pi\nu T} \frac{1}{4\pi\sigma^2} \\ &\quad \cdot \sum_{i,j} \iint_{B_{i,j}} \left[\exp\left(-\frac{|\vec{r}_{i,j}|^2}{4\nu T} - \frac{|\vec{x} - \vec{r}_{i,j}|^2}{4\sigma^2}\right) - \exp\left(-\frac{|\vec{s}|^2}{4\nu T} - \frac{|\vec{x} - \vec{s}|}{4\sigma^2}\right)\right] d\vec{s} \\ &= \frac{\gamma_0}{\sigma^2} \sum_{i,j} \exp\left(-\frac{|\vec{x} - \xi_{i,j}|}{4\sigma^2}\right) \left[O(r^3) + O\left(\frac{r^4}{\sigma^2}\right)\right] \\ &= \frac{\gamma_0}{\sigma^2} [O(l\sqrt{1-\alpha^2}) + O(1-\alpha^2)], \end{aligned} \tag{18}$$

where $\xi_{i,j} \in B_{i,j}$. The higher-order error terms from the DeMoivre–Laplace theorem are neglected because error terms from the Taylor expansion are of lower order.

The term $\frac{\gamma_0}{\sigma^2}$ will not be scaled explicitly in this analysis, but is worthy of further discussion. Vortex elements must be initialized on a length scale of $d = l^p$ where $p < 1$ to accurately approximate the initial conditions. For instance, Hald in his second convergence paper chooses $p = \frac{1}{2}$ to balance error terms in his convergence analysis [9]. For a general p, $\frac{\gamma_0}{\sigma^2} = O\left[l^{2(1-p)}\right]$. One could also use the new parameter and set $d = r$, in which case $\frac{\gamma_0}{\sigma^2} = O(1-\alpha^2)$. For any of these choices, the term $\frac{\gamma_0}{\sigma^2}$ is arbitrarily small as $l \to 0$ and $\alpha \to 1$. So, to make this analysis as general as possible, $\frac{\gamma_0}{\sigma^2}$ will not be scaled explicitly.

Finally, the approximate field can be compared with the exact solution.

$$\omega(\vec{x}) = \frac{\gamma_0}{4\pi(\nu T + \sigma^2)} \exp\left[-\frac{|\vec{x}|^2}{4\pi(\nu T + \sigma^2)}\right] + O(l\sqrt{1-\alpha^2}) + O(1-\alpha^2)$$

$$(19) \qquad = \frac{\gamma_0}{4\pi\nu T} \exp\left[-\frac{|\vec{x}|^2}{4\pi\nu T}\right] + \frac{\gamma_0}{\sigma^2}\left[O(l\sqrt{1-\alpha^2}) + O(1-\alpha^2) + O(l^2)\right].$$

These error estimates apply for all points in the plane, implying that these estimates are uniform. This gives rise to the following theorem.

THEOREM 2.1. *A single vortex element initially having variance $\alpha^2 l^2$ and circulation γ will induce a uniform field error no greater than*

$$C\frac{\gamma_0}{\sigma^2}[l\sqrt{(1-\alpha^2)} + (1-\alpha^2) + l^2],$$

where C depends on T only.

Since the cumulative refinement error can be made arbitrarily small, this refinement error can be interpreted as controllably small numerical disturbances to an ordinary vortex method with Gaussian vortex elements of width l. That is, given any tolerance, the refinement error can be made uniformly small for any core size, l. This being the case, the previous work of other investigators such as Hald or Majda can be applied to this core spreading method.

3. Linear convergence theory. This convergence theory has advantages and disadvantages when compared with existing theory. The major disadvantage is that the flow field $\vec{u}$ is specified and ω is computed as if it were a passive scalar, linearizing equation (1). The greatest advantage of this theory is that it establishes full uniform convergence in ω to the exact solution. While this investigation focuses on vorticity dynamics, ω could represent the concentration of any passive scalar. It is assumed that the flow velocity $\vec{u}$ and all its derivatives are known and bounded everywhere.

While initial conditions cannot be totally ignored, this study will not include a thorough investigation of the initial approximation. It is assumed that the exact initial conditions, $\omega(\vec{x}, 0)$, are known and at least piecewise continuous. The operator $I\omega(\vec{x}, t) = \omega(\vec{x}, 0)$ merely projects the vorticity field onto its initial values. Also, it is assumed there is some consistent algorithm for assigning circulations to the initial configuration of vortex elements. For instance, if the vortices are initially laid out in a grid, each vortex might have circulation equal to the $\omega(\vec{x}, 0)$ at that point multiplied by the area of each grid cell. This or any other consistent method for assigning initial circulations has the property that the circulations are bounded as $N \to \infty$ for the startup configuration. Since circulations do not grow with this method, they remain bounded for the duration of the simulation, $0 \leq t \leq T$.

At any time, each vortex element can be thought to occupy a given area, A_i, not necessarily related to the core width. At $t = 0$, this area might be used to assign initial circulations. As the flow evolves and elements spread and split, this area will move with the flow. Since the flow is incompressible, the A_i's will never grow in area though they may decrease or redistribute themselves due to refinement. Furthermore, since the flow derivatives are bounded, these regions will retain their shapes locally as N grows large and the areas grow proportionally smaller. Later, this will be used to justify converting sums to Riemann integrals.

In studying the convergence of this method, a residual operator emerges,

$$(20) \qquad R\omega = \partial_t\omega + (\vec{u} \cdot \nabla)\omega - \nu\nabla^2\omega.$$

If ω^* refers to the exact solution and $\widehat{\omega}$ refers to a computed solution, uniform convergence in ω means $\|\omega^* - \widehat{\omega}\|_\infty \to 0$ as $l \to 0$ and $\alpha \to 1$. Inspired by the work of Isaacson and Keller [12], convergence is divided into two separate conditions. First,

$$\|R\widehat{\omega}\| \to 0 \tag{21}$$

as $l \to 0$ and $\alpha \to 1$. (Obviously, $R\omega^* = 0$ always.) This is called "Condition A." It is not quite the same as consistency, though it does measure the local truncation error because local truncation error is a concept restricted to a discrete set of points usually on a regular grid. The other condition is that

$$\|\omega(t = T)\| \le K \int_0^T \|R\omega\| dt \tag{22}$$

for any ω is some class of functions where K is independent of l and α. This condition (Condition B) looks suspiciously like another convergence property called stability, but it is not at all the same. In fact, Condition B is solely a property of the governing equations and not the numerical method used to solve them. By combining these two results and letting $\omega = \omega^* - \widehat{\omega}$, linear convergence is established. While one is free to choose any norm for this formalism, this paper shall focus on the L^∞ norm.

3.1. Condition A. Conventional treatments of local truncation error are not applicable for a vortex method because it is impossible to substitute the exact solution into a difference equation representing the numerical scheme. Thus, traditional concepts such as consistency do not apply. This problem can be resolved by substituting a numerical solution into the residual operator R.

$$\begin{aligned}
\partial_t\widehat{\omega} &= \sum_{i=0}^{N} \gamma_i \partial_{\sigma^2}\left[\frac{1}{4\pi\sigma_i^2}\exp\left(-\frac{|\vec{x}-\vec{x}_i|^2}{4\sigma_i^2}\right)\right]\nu \\
&\quad + \sum_{i=0}^{N} \gamma_i \nabla_{\vec{x}_i}\left[\frac{1}{4\pi\sigma_i^2}\exp\left(-\frac{|\vec{x}-\vec{x}_i|^2}{4\sigma_i^2}\right)\right]\cdot\vec{u}(\vec{x}_i) \\
&= \sum_{i=0}^{N} \frac{\gamma_i}{4\pi\sigma_i^4}\left(\frac{|\vec{x}-\vec{x}_i|^2}{4\sigma_i^2} - 1\right)\exp\left(-\frac{|\vec{x}-\vec{x}_i|^2}{4\sigma_i^2}\right)\nu \\
&\quad + \sum_{i=0}^{N} \frac{\gamma_i(\vec{x}-\vec{x}_i)}{8\pi\sigma_i^4}\exp\left(-\frac{|\vec{x}-\vec{x}_i|^2}{4\sigma_i^2}\right)\cdot\vec{u}(\vec{x}_i), \\
\nabla\widehat{\omega} &= -\sum_{i=0}^{N} \frac{\gamma_i(\vec{x}-\vec{x}_i)}{8\pi\sigma_i^4}\exp\left(-\frac{|\vec{x}-\vec{x}_i|^2}{4\sigma_i^2}\right), \\
\nu\nabla^2\widehat{\omega} &= \sum_{i=0}^{N} \frac{\gamma_i}{4\pi\sigma_i^4}\left(\frac{|\vec{x}-\vec{x}_i|^2}{4\sigma_i^2} - 1\right)\exp\left(-\frac{|\vec{x}-\vec{x}_i|^2}{4\sigma_i^2}\right)\nu.
\end{aligned}$$

Combining these terms, one obtains an expression for the residual:

$$R\widehat{\omega} = \sum_{i=0}^{N} \frac{\gamma_i(\vec{x}-\vec{x}_i)}{8\pi\sigma_i^4}\exp\left(-\frac{|\vec{x}-\vec{x}_i|^2}{4\sigma_i^2}\right)\cdot[\vec{u}(\vec{x}_i) - \vec{u}(\vec{x})]. \tag{23}$$

Refinement-induced errors are not included here because they constitute arbitrarily small jump discontinuities in the time evolution. Therefore, it is more appropriate to incorporate these effects into Condition B in the next section.

From equation (23), it is possible to see why uncorrected core spreading fails to converge in most circumstances and why it will work in some special circumstances. Assuming particle trajectories are integrated exactly in time, the velocity deviations

$$(\vec{x} - \vec{x}_i) \cdot [\vec{u}(\vec{x}_i) - \vec{u}(\vec{x})] \simeq (\vec{x} - \vec{x}_i) D\vec{u}(\vec{x}_i)(\vec{x}_i - \vec{x})$$

across a vortex element centered at $\vec{x_i}$ are damped by the shape of the vortex element on the length scale equal to the core width. For CCSVM, this envelope scales on l while, for the uncorrected method, this envelope scales on $\sqrt{\nu t}$. As mentioned in §2, the core size for an uncorrected core spreading method is at least $\sqrt{\nu t}$. Therefore, it is impossible to make $R\widehat{\omega} \to 0$. Greengard presented this same result by finding that the uncorrected core spreading method converges to an equation different from Navier-Stokes [8]. In the case where flow deviations across every vortex element are orthogonal to the displacement from the center of the element, or, similarly, $D\vec{u}$ (the matrix of partial derivatives of $\vec{u}$) is strictly a rotation, this error is effectively zero.

EXAMPLE 2 (wherein core spreading does not need to be corrected). *To simulate the evolution of the vorticity field*

$$\frac{1}{4\pi} \exp\left(-\frac{x^2 + y^2}{4}\right),$$

the core spreading algorithm is employed using a single element with $\vec{x}_0 = \vec{0}$, $\sigma_0 = 1$, *and* $\gamma_0 = 1$. *Applying equations* (4)-(6), *the element will spread* ($\sigma_0 = 1 + \nu t$) *but not move. Furthermore, the induced flow across the single element is everywhere orthogonal to the displacement vector, hence the residual from equation* (23) *is exactly zero, corresponding to an exact solution to equation* (1).

However, a generic flow will not have the special property that flow deviations across the vorticity gradient will be zero, and so, will require corrections. Given that generic flows will require correction, core spreading corrected with adaptive spatial refinement will restrict the displacements over which velocity deviations can contribute to the residual. The extent to which refinement controls these deviations will reveal the dependence of the corrected method's accuracy on numerical parameters l and α.

Bounding $R\widehat{\omega}$ requires detailed analysis in many steps. First, equation (23) must be approximated by an integral. This integral is an anisotropic regularization of terms involving $\widehat{\omega}$ and the velocity field. This anisotropy is controlled by α while the smoothness of the regularization will depend on l. Finally, we found it necessary to include an artificial length scale, $d \simeq l^p$, to truncate the regularization, effectively introducing p into the convergence rate. Fortunately, p will only appear in higher-order terms and not affect the end result. Following this program without this artificial scale may be a valuable result in future investigations.

The first step in converting equation (23) into an integral is to extend the σ_i into a function $\sigma(\vec{x})$ such that $\alpha l \leq \sigma(\vec{x}) \leq l$ and $\sigma(\vec{x}_i) = \sigma_i$. There are many ways to construct this function. One might be to consider that each vortex blob position is contained in a small region, A_i, as discussed earlier, and $\sigma(\vec{x})$ is piecewise constant on these regions. Similarly, $\widehat{\widehat{\omega}}(\vec{x}) = \frac{\gamma_i}{|A_i|}$ on A_i.

In the limit $l \to 0$, the function $\widehat{\widehat{\omega}}(\vec{x}) \to \widehat{\omega}(\vec{x})$ almost everywhere. Unless otherwise noted, all integrals are evaluated over the entire real line. There are several relationships in addition to equation (4) which relate $\widehat{\omega}$ to convolutions.

$$\left\| \widehat{\omega}(\vec{x}) - \iint \frac{\widehat{\omega}(\vec{s})}{4\pi l^2} \exp\left(-\frac{|\vec{x} - \vec{s}|^2}{4l^2}\right) d\vec{s} \right\|_\infty = O(\|\nabla\widehat{\omega}\|, l),$$

$$\left\| \sum_{i=1}^{N} \frac{\gamma_i}{4\pi\sigma_i^2} \exp\left(-\frac{|\vec{x}-\vec{x}_i|^2}{4\sigma_i^2}\right) - \iint \frac{\widehat{\omega}(\vec{s})}{4\pi\sigma^2(\vec{s})} \exp\left(-\frac{|\vec{x}-\vec{s}|^2}{4\sigma^2(\vec{s})}\right) d\vec{s} \right\|_\infty = O\left(\frac{1}{\sqrt{N}}\right).$$

Thus,

$$\begin{aligned} &\left\| \iint \frac{\widehat{\omega}(\vec{s})}{4\pi l^2} \exp\left(-\frac{|\vec{x}-\vec{s}|^2}{4l^2}\right) - \frac{\widehat{\omega}(\vec{s})}{4\pi\sigma^2(\vec{s})} \exp\left(-\frac{|\vec{x}-\vec{s}|^2}{4\sigma^2(\vec{s})}\right) d\vec{s} \right\|_\infty \\ &\qquad = O\left(\frac{1}{\sqrt{N}}\right) + O(\|\nabla\widehat{\omega}\|, l). \end{aligned} \tag{24}$$

Now,

$$\begin{aligned} &\frac{\widehat{\omega}(\vec{s})}{4\pi\sigma^2(\vec{s})} \exp\left(-\frac{|\vec{x}-\vec{s}|^2}{4\sigma^2(\vec{s})}\right) d\vec{s} \\ &\qquad = \frac{\widehat{\omega}(\vec{s})}{4\pi l^2} \left[1 + \frac{l^2-\sigma^2(\vec{s})}{\sigma^2(\vec{s})}\right] \exp\left(-\frac{|\vec{x}-\vec{s}|^2}{4l^2}\left[1 + \frac{l^2-\sigma^2(\vec{s})}{\sigma^2(\vec{s})}\right]\right) d\vec{s}. \end{aligned} \tag{25}$$

However, the latter term in the exponential is close to unity when $|\vec{x}-\vec{s}|$ is close to zero.

$$\left| \exp\left[-\frac{|\vec{x}-\vec{s}|^2}{4l^2}\left(\frac{l^2-\sigma^2(\vec{s})}{\sigma^2(\vec{s})}\right)\right] - 1 \right| \le \frac{|\vec{x}-\vec{s}|^2}{4l^2}\left(\frac{l^2-\sigma^2(\vec{s})}{\sigma^2(\vec{s})}\right). \tag{26}$$

Thus,

$$\begin{aligned} &\left\| \left(\iint_{\vec{s}<d} + \iint_{\vec{s}>d}\right) \frac{\widehat{\omega}(\vec{s})}{4\pi l^2} \exp\left(-\frac{|\vec{x}-\vec{s}|^2}{4l^2}\right) - \frac{\widehat{\omega}(\vec{s})}{4\pi\sigma(\vec{s})^2} \exp\left(-\frac{|\vec{x}-\vec{s}|^2}{4\sigma(\vec{s})^2}\right) d\vec{s} \right\|_\infty \\ &\le \left\| \iint_{\vec{s}<d} \frac{\widehat{\omega}(\vec{s})}{4\pi l^2} \exp\left(-\frac{|\vec{x}-\vec{s}|^2}{4l^2}\right) - \frac{\widehat{\omega}(\vec{s})}{4\pi l^2} \left[1 + \frac{l^2-\sigma^2(\vec{s})}{\sigma^2(\vec{s})}\right] \exp\left(-\frac{|\vec{x}-\vec{s}|^2}{4\sigma(\vec{s})^2}\right) d\vec{s} \right\| \\ &\quad + \|\widehat{\omega}\| \left(1+\alpha^{-2}\right) \exp\left[-\frac{1}{4}\left(\frac{d}{l}\right)^2\right] \\ &\le \frac{\|\widehat{\omega}\|}{4\pi l^2} \frac{1-\alpha^2}{\alpha^2} \iint_{\vec{s}<d} \left(1 + \frac{|\vec{x}-\vec{s}|^2}{4l^2}\right) \exp\left(-\frac{|\vec{x}-\vec{s}|^2}{4l^2}\right) d\vec{s} \\ &\quad + \|\widehat{\omega}\| \left(1+\alpha^{-2}\right) \exp\left[-\frac{1}{4}\left(\frac{d}{l}\right)^2\right] \\ &\le \|\widehat{\omega}\| \frac{1-\alpha^2}{\alpha^2} \left\{3 + \left[3 - \left(\frac{d}{l}\right)^2\right] \exp\left[-\frac{1}{4}\left(\frac{d}{l}\right)^2\right]\right\} \\ &\quad + \|\widehat{\omega}\| \left(1+\alpha^{-2}\right) \exp\left[-\frac{1}{4}\left(\frac{d}{l}\right)^2\right]. \end{aligned} \tag{27}$$

Allowing $d = l^p$ where $p < 1$, the exponential terms decay faster than any polynomial in l or α. So,

$$\begin{aligned} &\left\| \left(\iint_{\vec{s}<d} + \iint_{\vec{s}>d}\right) \frac{\widehat{\omega}(\vec{s})}{4\pi l^2} \exp\left(-\frac{|\vec{x}-\vec{s}|^2}{4l^2}\right) - \frac{\widehat{\omega}(\vec{s})}{4\pi\sigma(\vec{s})^2} \exp\left(-\frac{|\vec{x}-\vec{s}|^2}{4\sigma(\vec{s})^2}\right) d\vec{s} \right\|_\infty \\ &\qquad = O\left(1-\alpha^2\right). \end{aligned}$$

Therefore,

$$\left\| \iint \frac{\widehat{\omega}(\vec{s}) - \widehat{\widehat{\omega}}(\vec{s})}{4\pi l^2} \exp\left(-\frac{|\vec{x}-\vec{s}|^2}{4l^2}\right) d\vec{s} \right\|_\infty$$

$$= O\left(\frac{1}{\sqrt{N}}\right) + O(\|\nabla\widehat{\omega}\|, l) + O(1-\alpha^2). \tag{28}$$

This can be extended to the residual because the function

$$[\vec{u}(\vec{x}) - \vec{u}(\vec{s})] \cdot (\vec{s} - \vec{x})$$

is locally integrable. Hence,

$$\begin{aligned} R\widehat{\omega} &= \iint \frac{\widehat{\widehat{\omega}}(\vec{s})}{8\pi\sigma^4(\vec{s})} \exp\left[-\frac{|\vec{x}-\vec{s}|^2}{4\sigma^2(\vec{s})}\right] [\vec{u}(\vec{x}) - \vec{u}(\vec{s})] \cdot (\vec{s}-\vec{x}) d\vec{s} + O\left(\frac{1}{\sqrt{N}}\right) \\ &= \iint \frac{\widehat{\omega}(\vec{s})}{8\pi\sigma^4(\vec{s})} \exp\left[-\frac{|\vec{x}-\vec{s}|^2}{4\sigma^2(\vec{s})}\right] [\vec{u}(\vec{x}) - \vec{u}(\vec{s})] \cdot (\vec{s}-\vec{x}) d\vec{s} \\ &\quad + O\left(\frac{1}{\sqrt{N}}\right) + O(\|\nabla\widehat{\omega}\|, l) + O(1-\alpha^2). \end{aligned} \tag{29}$$

While $\|\widehat{\omega}\|$ and $\|\nabla\widehat{\omega}\|$ are bounded initially, nothing in this analysis asserts that their evolution is well behaved in the limit as $\alpha \to 1$ and $l \to 0$. However, similar analysis verifies that these two quantities are bounded as the numerical method converges. These issues are addressed in the appendices. Thus, for the rest of this section, it is assumed that the vorticity field and its gradients are bounded as the numerical parameters approach their limiting values.

Now, it is possible to analyze the residual operator

$$\begin{aligned} \|R\widehat{\omega}\|_\infty &= \left\| \left(\iint_{|\vec{s}|<d, |\vec{s}|>d} \right) \frac{\widehat{\omega}(\vec{s})}{8\pi\sigma^4(\vec{s})} \exp\left[-\frac{|\vec{x}-\vec{s}|^2}{4\sigma^2(\vec{s})}\right] [\vec{u}(\vec{x}) - \vec{u}(\vec{s})] \cdot (\vec{s}-\vec{x}) d\vec{s} \right\| \\ &\quad + O\left(\frac{1}{\sqrt{N}}\right) + O(\|\nabla\widehat{\omega}\|, l) + O(1-\alpha^2). \end{aligned} \tag{30}$$

The contributions to the residual are localized about $\vec{x}$ and the effects from $\vec{s} > d$ decay exponentially. Furthermore, the integrand for $\vec{x} < d$ can be decomposed into isotropic ($\sigma = l$) and anisotropic (σ varies in space) parts outside of the exponential. The expression $D\vec{u}$ is the matrix of partial derivatives of $\vec{u}$ and $\|D\vec{u}\|$ is the induced operator norm on L^∞.

$$\begin{aligned} \|R\widehat{\omega}\|_\infty \leq &\left\| \iint_{|\vec{s}|<d} \frac{\widehat{\omega}(\vec{s})}{8\pi l^4} \exp\left[-\frac{|\vec{x}-\vec{s}|^2}{4\sigma^2(\vec{s})}\right] [\vec{u}(\vec{x}) - \vec{u}(\vec{s})] \cdot (\vec{s}-\vec{x}) d\vec{s} \right\| \\ &+ \left\| \iint_{|\vec{s}|<d} \frac{\widehat{\omega}(\vec{s})}{8\pi l^4} \left(\frac{l^4 - \sigma^4(\vec{s})}{\sigma^4(\vec{s})}\right) \exp\left[-\frac{|\vec{x}-\vec{s}|^2}{4\sigma^2(\vec{s})}\right] [\vec{u}(\vec{x}) - \vec{u}(\vec{s})] \cdot (\vec{s}-\vec{x}) d\vec{s} \right\| \\ &+ \frac{\|\widehat{\omega}\| \, \|D\vec{u}\|}{2\alpha^4} \left[\left(\frac{d}{l}\right)^2 + 4\right] \exp\left[-\frac{1}{4}\left(\frac{d}{l}\right)^2\right] \\ &+ O\left(\frac{1}{\sqrt{N}}\right) + O(\|\nabla\widehat{\omega}\|, l) + O(1-\alpha^2). \end{aligned} \tag{31}$$

The localization of the integral justifies a Taylor expansion of the vorticity field within the first term of equation (31). The second term can be bounded in terms of the vorticity field and the flow deviations.

$$
\begin{aligned}
&\|R\widehat{\omega}\|_\infty \\
&\quad\leq \left\| \iint_{|\vec{s}|<d} \frac{\widehat{\omega}(\vec{x}) + \nabla\widehat{\omega}(\vec{\xi}) \cdot (\vec{s}-\vec{x})}{8\pi l^4} \exp\left[-\frac{|\vec{x}-\vec{s}|^2}{4\sigma^2(\vec{s})}\right] \left[\vec{u}(\vec{x}) - \vec{u}(\vec{s})\right] \cdot (\vec{s}-\vec{x}) d\vec{s} \right\| \\
&\qquad + \|\widehat{\omega}\| \, \|D\vec{u}\| \frac{1}{8\pi l^4} \frac{1-\alpha^4}{\alpha^4} \iint_{|\vec{s}|<d} |\vec{x}-\vec{s}|^2 \exp\left(-\frac{|\vec{s}-\vec{x}|^2}{4l^2}\right) d\vec{s} \\
&\qquad + \frac{\|\widehat{\omega}\| \, \|D\vec{u}\|}{2\alpha^4} \left[\left(\frac{d}{l}\right)^2 + 4\right] \exp\left[-\frac{1}{4}\left(\frac{d}{l}\right)^2\right] \\
&\qquad + O\left(\frac{1}{\sqrt{N}}\right) + O(\|\nabla\widehat{\omega}\|, l) + O(1-\alpha^2). \qquad (32)
\end{aligned}
$$

Within the first term of equation (31), the exponential can also be decomposed into isotropic and anisotropic parts. The anisotropic exponential is close to unity when α is close to unity as expected.

$$
\begin{aligned}
\|R\widehat{\omega}\|_\infty \leq &\left\| \iint_{|\vec{s}|<d} \frac{\widehat{\omega}(\vec{x})}{8\pi l^4} \exp\left[-\frac{|\vec{x}-\vec{s}|^2}{4l^2}\right] \right. \\
&\left. \times \exp\left[-\frac{|\vec{x}-\vec{s}|^2}{4l^2}\left(\frac{l^2-\sigma^2(\vec{s})}{\sigma^2(\vec{s})}\right)\right] \left[\vec{u}(\vec{x}) - \vec{u}(\vec{s})\right] \cdot (\vec{s}-\vec{x}) d\vec{s} \right\| \\
&+ \left\| \iint_{|\vec{s}|<d} \frac{\nabla\widehat{\omega}(\vec{\xi}) \cdot (\vec{s}-\vec{x})}{8\pi l^4} \exp\left[-\frac{|\vec{x}-\vec{s}|^2}{4\sigma^2(\vec{s})}\right] \left[\vec{u}(\vec{x}) - \vec{u}(\vec{s})\right] \cdot (\vec{s}-\vec{x}) d\vec{s} \right\| \\
&+ \|\widehat{\omega}\| \, \|D\vec{u}\| \frac{1-\alpha^4}{2\alpha^4} \left\{4 - \left[\left(\frac{d}{l}\right)^2 + 4\right] \exp\left[-\frac{1}{4}\left(\frac{d}{l}\right)^2\right]\right\} \\
&+ \frac{\|\widehat{\omega}\| \, \|D\vec{u}\|}{2\alpha^4} \left[\left(\frac{d}{l}\right)^2 + 4\right] \exp\left[-\frac{1}{4}\left(\frac{d}{l}\right)^2\right] \\
&+ O\left(\frac{1}{\sqrt{N}}\right) + O(\|\nabla\widehat{\omega}\|, l) + O(1-\alpha^2). \qquad (33)
\end{aligned}
$$

The anisotropic exponential in the first term of equation (33) can be approximated by 1 plus a small error which is bounded by the second term of its Taylor expansion.

$$
\begin{aligned}
\|R\widehat{\omega}\|_\infty \leq &\left\| \iint_{|\vec{s}|<d} \frac{\widehat{\omega}(\vec{x})}{8\pi l^4} \exp\left[-\frac{|\vec{x}-\vec{s}|^2}{4l^2}\right] \left[\vec{u}(\vec{x}) - \vec{u}(\vec{s})\right] \cdot (\vec{s}-\vec{x}) d\vec{s} \right\| \\
&+ \|\widehat{\omega}\| \, \|D\vec{u}\| \frac{d^2(1-\alpha^2)}{8\pi\alpha^4 l^6} \iint_{|\vec{s}|<d} |\vec{s}-\vec{x}|^2 \exp\left[-\frac{|\vec{x}-\vec{s}|^2}{4l^2}\right] d\vec{s} \\
&+ \frac{\|\nabla\widehat{\omega}\| \, \|D\vec{u}\|}{8\pi l^4} \iint_{|\vec{s}|<d} |\vec{s}-\vec{x}|^3 \exp\left[-\frac{|\vec{x}-\vec{s}|^2}{4l^2}\right] d\vec{s} \\
&+ \frac{\|\widehat{\omega}\| \, \|D\vec{u}\|}{2\alpha^4} \left\{4(1+\alpha^2)(1-\alpha^2) + \alpha^4 \left[\left(\frac{d}{l}\right)^2 + 4\right] \exp\left[-\frac{1}{4}\left(\frac{d}{l}\right)^2\right]\right\} \\
&+ O\left(\frac{1}{\sqrt{N}}\right) + O(\|\nabla\widehat{\omega}\|, l) + O(1-\alpha^2). \qquad (34)
\end{aligned}
$$

Finally, the velocity field is linearized about $\vec{x}$ in the first term. The remainder is bounded in terms of $H\vec{u}$, the Hessian.

$$
\begin{aligned}
\|R\widehat{\omega}\|_\infty \leq{}& \left\| \iint_{|\vec{s}|<d} \frac{\widehat{\omega}(\vec{x})}{8\pi l^4} \exp\left[-\frac{|\vec{x}-\vec{s}|^2}{4l^2}\right] \left[D\vec{u}(\vec{x})(\vec{x}-\vec{s})\right]\cdot(\vec{s}-\vec{x})d\vec{s} \right\| \\
&+ \frac{\|\widehat{\omega}\|\,\|H\vec{u}\|}{8\pi l^4} \iint_{|\vec{s}|<d} \exp\left[-\frac{|\vec{x}-\vec{s}|^2}{4l^2}\right] |\vec{s}-\vec{x}|^3 d\vec{s} \\
&+ \|\widehat{\omega}\|\,\|D\vec{u}\| \frac{(1-\alpha^2)}{2\alpha^4}\left(\frac{d}{l}\right)^2 \left\{4 - \left[\left(\frac{d}{l}\right)^2+4\right]\exp\left[-\frac{1}{4}\left(\frac{d}{l}\right)^2\right]\right\} \\
&+ \frac{3\|\nabla\widehat{\omega}\|\,\|D\vec{u}\|}{2} \left\{2\sqrt{\pi}l - \left[\frac{d^3}{l^2}\right]\exp\left[-\frac{1}{4}\left(\frac{d}{l}\right)^2\right]\right. \\
&\qquad \left. - 2d\exp\left[-\frac{1}{4}\left(\frac{d}{l}\right)^2\right] - 2\sqrt{\pi}l\exp\left[-\frac{1}{8}\left(\frac{d}{l}\right)^2\right]\right\} \\
&+ \frac{\|\widehat{\omega}\|\,\|D\vec{u}\|}{2\alpha^4} \left\{4(1+\alpha^2)(1-\alpha^2) + \alpha^4\left[\left(\frac{d}{l}\right)^2+4\right]\exp\left[-\frac{1}{4}\left(\frac{d}{l}\right)^2\right]\right\} \\
&+ O\left(\frac{1}{\sqrt{N}}\right) + O(\|\nabla\widehat{\omega}\|, l) + O(1-\alpha^2).
\end{aligned}
\tag{35}
$$

Since the fluid is incompressible, $D\vec{u}$ has zero trace. Therefore, the first term of equation (35) is zero.

$$
\begin{aligned}
\|R\widehat{\omega}\|_\infty \leq{}& \frac{3\|\widehat{\omega}\|\,\|H\vec{u}\|}{2} \left\{2\sqrt{\pi}l - \left[\frac{d^3}{l^2}\right]\exp\left[-\frac{1}{4}\left(\frac{d}{l}\right)^2\right]\right. \\
&\qquad \left. -2d\exp\left[-\frac{1}{4}\left(\frac{d}{l}\right)^2\right] - 2\sqrt{\pi}l\exp\left[-\frac{1}{8}\left(\frac{d}{l}\right)^2\right]\right\} \\
&+ \|\widehat{\omega}\|\,\|D\vec{u}\| \frac{(1-\alpha^2)}{2\alpha^4}\left(\frac{d}{l}\right)^2 \left\{4 - \left[\left(\frac{d}{l}\right)^2+4\right]\exp\left[-\frac{1}{4}\left(\frac{d}{l}\right)^2\right]\right\} \\
&+ \frac{3\|\nabla\widehat{\omega}\|\,\|D\vec{u}\|}{2} \left\{2\sqrt{\pi}l - \left[\frac{d^3}{l^2}\right]\exp\left[-\frac{1}{4}\left(\frac{d}{l}\right)^2\right]\right. \\
&\qquad \left. -2d\exp\left[-\frac{1}{4}\left(\frac{d}{l}\right)^2\right] - 2\sqrt{\pi}l\exp\left[-\frac{1}{8}\left(\frac{d}{l}\right)^2\right]\right\} \\
&+ \frac{\|\widehat{\omega}\|\,\|D\vec{u}\|}{2\alpha^4} \left\{4\left[1+\alpha^2+\left(\frac{d}{l}\right)^2\right](1-\alpha^2)\right. \\
&\qquad \left. + \left[\alpha^4 - \left(\frac{d}{l}\right)^2(1-\alpha^2)\right]\left[\left(\frac{d}{l}\right)^2+4\right]\exp\left[-\frac{1}{4}\left(\frac{d}{l}\right)^2\right]\right\} \\
&+ O\left(\frac{1}{\sqrt{N}}\right) + O(\|\nabla\widehat{\omega}\|, l) + O(1-\alpha^2).
\end{aligned}
\tag{36}
$$

If $d = l^p$ where $p < 1$ then all but a few of these terms decay faster than any polynomial in l as $l \to 0$. The remaining terms are the true rate of uniform convergence for the corrected core spreading vortex method.

$$\|R\widehat{\omega}\|_\infty \le O\left(\frac{1}{\sqrt{N}}\right) + O(\|\widehat{\omega}\|, l) + O(\|\nabla\widehat{\omega}\|, l) + O[\|\widehat{\omega}\|, (1-\alpha^2)]$$

$$+O[\|\widehat{\omega}\|, (1-\alpha^2), l^{2(p-1)}]. \tag{37}$$

The artificial length scale d only enters into the fourth term which is of higher order than the previous three. While $\|\widehat{\omega}\|$ and $\|\nabla\widehat{\omega}\|$ are bounded initially, nothing in this analysis asserts that their evolution is well behaved in the limit as $\alpha \to 1$ and $l \to 0$. However, analysis of the growth of $\widehat{\omega}$ and $\nabla\widehat{\omega}$, similar to that of $R\widehat{\omega}$, proves that this is the case [15]. In this analysis, the growth of $\widehat{\omega}$ depends on $\nabla\widehat{\omega}$, but closure is achieved for the estimates of $\nabla\widehat{\omega}$. This establishes the following theorem.

THEOREM 3.1 (Condition A). *If $\widehat{\omega}$ is a solution to equations* (4), (5), *and* (6) *with a suitably stable refinement scheme to maintain core sizes between αl and l then the residual operator is at most $O(l) + O(1-\alpha^2)$ for large values of N.*

3.2. Condition B. In this section, it will be shown that any function can be uniformly bounded in terms of its response to the operator R. To facilitate this result, a new operator is defined:

$$L(t)f(\vec{x}(t), t) = -\nu \int_0^t \nabla^2 f(\vec{x}(s), s)ds, \tag{38}$$

where $\vec{x}$ is the position of a Lagrangian particle moving with a velocity $\vec{u}(\vec{x}(t), t)$. If $L(t)$ is restricted to a class of functions such that the function and all its derivatives are smaller than M, then $L(t)$ is bounded by νMT in the operator norm induced by L_∞.

By establishing a relationship between ω and $R\omega$ along Lagrangian trajectories, it is possible to obtain a uniform estimate. Since $\widehat{\omega}$ does not have classical time derivatives during refinement, we shall separate it into two parts:

$$\delta\widehat{\omega} = \widehat{\omega} - \widehat{\omega}^+, \tag{39}$$

where $\delta\widehat{\omega}$ is the difference between unrefined and refined $\widehat{\omega}$, and $\widehat{\omega}^+$ is the smoothly evolving difference between the two. Though $\delta\widehat{\omega}$ is not continuous in time, Theorem 2.1 establishes that the time integral of this quantity is uniformly bounded by quantities of order l^2, $1-\alpha^2$ and $l\sqrt{1-\alpha^2}$, all of which are of higher order than the error terms in $R\widehat{\omega}^+$. Thus, one only expects to apply Condition B to $\widehat{\omega}^+$ and ω.

$$\begin{aligned}
\omega(\vec{x}(T), T) &= e^{-L(T)}e^{L(T)}\omega(\vec{x}(T), T) \\
&= e^{-L(T)}\left[\int_0^T \frac{d}{dt}e^{L(t)}\omega(\vec{x}(t), t)dt + \omega(\vec{x}(0), 0)\right] \\
&= e^{-L(T)}\left[\int_0^T \frac{d}{dt}e^{L(t)}\omega(\vec{x}(t), t)dt + I\omega(\vec{x})\right] \\
&= e^{-L(T)}\left\{\int_0^T e^{L(t)}\left[\frac{D\omega}{Dt}(\vec{x}(t), t) - \nu\nabla^2\omega(\vec{x}(t), t)\right]dt + I\omega(\vec{x})\right\} \\
&= e^{-L(T)}\left[\int_0^T e^{L(t)}R\omega(\vec{x}(t), t)dt + I\omega(\vec{x})\right] \\
&= \int_0^T e^{L(t)-L(T)}R\omega(\vec{x}(t), t)dt + e^{-L(T)}I\omega(\vec{x}).
\end{aligned} \tag{40}$$

Thus, for any initial position,

$$|\omega(\vec{x}(T),T)| \leq e^{\nu MT}\left[\int_0^T |R\omega(\vec{x}(t),t)|dt + |I\omega(\vec{x})|\right]. \tag{41}$$

This pointwise estimate can be extended to yield the desired result.

THEOREM 3.2 (Condition B). *For any function ω with two continuous derivatives,*

$$\|\omega(\vec{x}(T),T)\|_\infty \leq e^{\nu MT}\left[\int_0^T \|R\omega(\vec{x}(t),t)\|_\infty dt + \|I\omega(\vec{x})\|_\infty\right].$$

3.3. *A* + *B* = Convergence. Combining Theorems 3.1 and 3.2, one obtains a uniform convergence estimate. From Theorem 3.2 and the linearity of R and I,

$$\begin{aligned}&\|\widehat{\omega}(\vec{x}(T),T) - \omega^*(\vec{x}(T),T)\|_\infty\\ &\quad\leq e^{\nu MT}\left[\int_0^T \|R\widehat{\omega}^+(\vec{x}(t),t) - R\omega^*(\vec{x}(t),t)\|_\infty dt + \|I\widehat{\omega}(\vec{x}) - I\omega^*(\vec{x})\|_\infty\right]\\ &\qquad+\|\delta\widehat{\omega}(T)\|_\infty.\end{aligned} \tag{42}$$

But, Theorem 3.1 asserts

$$\begin{aligned}&\|\widehat{\omega}(\vec{x}(T),T) - \omega^*(\vec{x}(T),T)\|_\infty\\ &\quad\leq e^{\nu MT}\left\{\int_0^T \left[O(l) + O(1-\alpha^2)\right]dt + \|I\widehat{\omega}(\vec{x}) - I\omega^*(\vec{x})\|_\infty\right\} + \|\delta\widehat{\omega}(T)\|_\infty.\end{aligned} \tag{43}$$

Since initially, no refinement has occurred, $\widehat{\omega}^+ = \widehat{\omega}$ at time zero. If refinement did occur initially, any error would be incorporated into $I\widehat{\omega}$ forming a new initial error, assuming that one is using a convergent scheme for approximating the initial conditions which is at least $O(l)$.

THEOREM 3.3 (convergence of improved core spreading method). *Assuming there existed an exact solution to equation* (1) *which is twice differentiable, the solution to equations* (4), (5), *and* (6) (*where $\vec{u}$ corresponds to the exact solution*) *with a suitable refinement algorithm will converge to this exact solution and*

$$\|\widehat{\omega}(\vec{x}(T),T) - \omega^*(\vec{x}(T),T)\|_\infty \leq O(l) + O(1-\alpha^2)$$

for $t \leq T$.

While this convergence rate appears modest, it is stronger than previous results in many ways. Previous findings are limited to velocity fields at particle positions. Convergence of the vorticity field itself is substantially weaker, and again, is only measured at particle positions. The above analysis determines a rate of convergence which is applicable to the vorticity field itself in the far stronger uniform norm. Of course, this result can be extended to the velocity field without loss of accuracy. This result will be verified experimentally in §5.

4. Efficiency and fusion. Since the computational complexity of a vortex method is $O(N^2)$, the feasibility of computing a flow to a prescribed accuracy hinges directly on the problem size, N. Clearly, two blobs sharing the same centroid and width can be merged into a single blob with no induced error in the vorticity field. "Vortex fusion" shall refer to the general process of merging many overlapping basis functions. The necessity of adaptive spatial refinement, discussed earlier, does not imply the existence of a fusion process. Rather, they are two separate mechanisms for solving two separate problems: one corrects consistency error while the other reduces the problem size. This section presents the basic theorems describing the uniform estimate for the induced fusion error without proof, leaving the details of this analysis to [16]. From these results, one will see that it is not necessary to rigorously examine all possible subsets of a configuration of vortex elements because the uniform estimate does not depend on the number of elements merged. Rather, it only depends on certain nondimensional parameters related to the separation and width of vortex elements in the simulation.

There are two error control parameters for vortex fusion based on a direct comparison between each individual vortex in the original group and the fused vortex. To begin, the vortices to be fused are labeled from 1 to n, where $n \leq N$, and have distinct circulations, widths, and positions. The post-fusion element shall be labeled 0. Also, it is necessary to assume that all γ_i's have the same sign for reasons discussed in [16]. The circulation, position, and width of the fused vortex is

$$\gamma_0 = \sum_{i=n}^{N} \gamma_i, \tag{44}$$

$$\vec{x}_0 = \sum_{i=n}^{N} \gamma_i \vec{x}_i, \tag{45}$$

$$4\gamma_0\sigma_0^2 = \sum_{i=n}^{N} \gamma_i(4\sigma_i^2 + |\vec{x}_i|^2). \tag{46}$$

For convenience, all positions will be translated to the center of mass of the original cluster of elements so that $\vec{x}_0 = 0$. Equation (46) constrains the fused blob width but does not play a key role in the uniform error bound, as do the others. Therefore, it may be possible to use other constraints in place of this one. The pointwise field error is

$$e(\vec{x}) = \frac{\gamma_0}{4\pi\sigma_0^2}\exp\left(-\frac{|\vec{x}|^2}{4\sigma_0^2}\right) - \sum_{i=1}^{n}\frac{\gamma_i}{4\pi\sigma_i^2}\exp\left(-\frac{|\vec{x}-\vec{x}_i|^2}{4\sigma_i^2}\right). \tag{47}$$

By isolating the appropriate length scales in the problem and studying the resulting expressions, it is possible to establish the following theorem.

THEOREM 4.1 (fusion error bound). *Suppose n vortex elements are fused into a single element with circulation, position, and width defined by equations* (44), (45), *and* (46). *Furthermore, assume that all circulations have the same sign and that*

$$\left(\frac{|\vec{x}_i|}{2\sigma_0}\right)^2 \leq R$$

and

$$b_1 \leq \left(\frac{\sigma_0}{\sigma_i}\right)^2 \leq b_2$$

for each vortex. Then, the uniform induced field error, $\|e(\hat{x})\|_\infty$, is no greater than

$$\frac{|\gamma_0|M}{4\pi\sigma_0^2},$$

where M is the maximum value of

$$\exp\left(-|\hat{x}|^2\right) - \frac{\sigma_0^2}{\sigma_i^2}\exp\left[-|\hat{x}-\hat{x}_i|^2\left(\frac{\sigma_0^2}{\sigma_i^2}\right)\right]$$

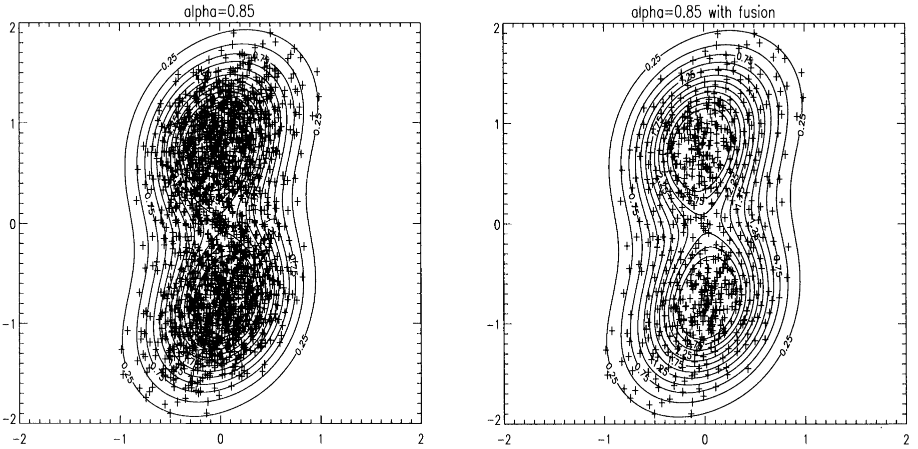


FIG. 3. *A demonstration of the efficacy of fusion. Vorticity field with element positions at* $t = 2.5$ *for* $\alpha = 0.85$ *in the experiment described in* §5.2 *without fusion* (*left*) *and with fusion* (*right*). *The absolute pointwise error between these two fields is less than* 0.06%, *while the problem size has been diminished by a factor of almost* 3.

restricted to the compact set of parameter space described by R, b_1, *and* b_2. *Thus,* $M \equiv M(R, b_1, b_2)$.

To find the value of M, it is not necessary to search the entire space $[0, R] \times [b_1, b_2]$ because further analysis reveals that these maximal values are only attained on two corners of this rectangle in parameter space. This is summarized in the second and final theorem describing these merger estimates.

THEOREM 4.2 (explicit determination of fusion error bound). *If*

$$\left(\frac{|\vec{x}_i|}{2\sigma_0}\right)^2 \leq R$$

and

$$b_1 \leq \left(\frac{\sigma_0}{\sigma_i}\right)^2 \leq b_2,$$

then the absolute maximal value of the uniform fusion error function

$$\exp\left(-|\hat{x}|^2\right) - \frac{\sigma_0^2}{\sigma_i^2} \exp\left[-|\hat{x} - \hat{x}_i|^2 \left(\frac{\sigma_0^2}{\sigma_i^2}\right)\right]$$

is achieved with parameter values of either (R, b_1) *or* (R, b_2). *Furthermore, this absolute maximum is attained when* $\vec{x}$ *is colinear with the origin and* $\vec{x}_i$.

The consequence of these theorems is that any number of same-signed vortex elements can be fused together provided they have sufficiently similar core widths and are in close proximity to each other relative to their core sizes. Furthermore, values of M can be tabulated easily by evaluating the function in Theorem 4.2 at two points. Together, these two results describe a fusion algorithm that can reduce the problem size of a given simulation [16]. Figure 3 presents a demonstration of this reduction for a simulation of two merging patches of vorticity discussed more thoroughly in §5.2.

5. Simulations, measurements, and comparisons. In this section, the new method is used to simulate two different flows. The first flow, a simple vortex patch, has an exact

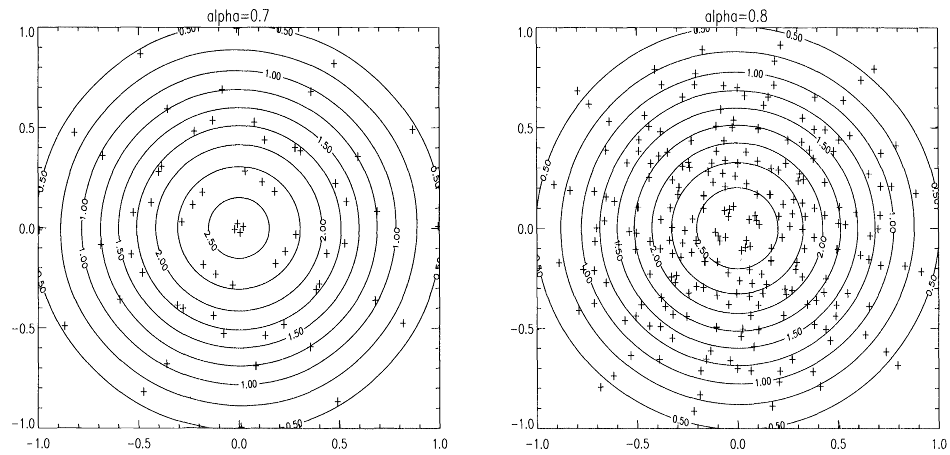


FIG. 4. *Vorticity field of a radially symmetric patch of vorticity with element positions at* $t = 1.0$ *for* $\alpha = 0.7$ *and* $\alpha = 0.8$.

analytic solution, and so can be used for a precise convergence study. Also, it can be argued that generic flows are composed of many simple vortex patches so that its performance in the simpler case can lend insight into its performance in more complicated settings. The second flow is more complicated but is also a good testing ground for a convergence study. As a second measurement of the method's performance, the results of CCSVM are analyzed to see whether or not they conserve certain integral invariants. Finally, one can directly compare CCSVM and RWVM by using them to solve the same problem on the same machine with the same total number of elements. As mentioned in §2.2, CCSVM and RWVM are equivalent in the limit as problem sizes grow infinitely large. However, this is never the case when solving a problem, so it is most meaningful to compare the two methods at moderate problem sizes.

5.1. A radially symmetric spreading patch of vorticity. The first example concerns the evolution of a single Gaussian blob. While seeming simple on the surface, this example could be said to be of fundamental importance because the analytical solution is known. Using this analytical solution, one can measure the uniform field error in numerical simulations to demonstrate the linear uniform convergence estimate from the previous section. Also, this simple example will demonstrate that refinement is a stable and locally adaptive process similar to the local regridding scheme of Hou, Lowengrub, and Shelley [11].

The initial field is expressed by a single vortex element ($\gamma = 5.0$, $\sigma = 0.2$) centered at the origin. The numerical parameter l is fixed at a reasonably small value of 0.25, and α is allowed to vary. The vortex positions and widths were updated using forward Cauchy-Euler with a small enough time step so that no variations in element positions were observed at the resolution of our printer. The flow was simulated with $T = 1$ and a large viscosity, $\nu = 0.1$, so that both convective and diffusive effects would be important. (Indeed, during the course of this simulation, the width of the exact solution nearly doubles.) In Fig. 4, one can see how the vortex elements split and adapt across streamlines to accurately express the solution with a length scale at or below l.

Earlier investigators noted that vortex methods for Euler equations lose accuracy after a finite period of time because flow deviations separate vortex elements until they cease to overlap. Hou, Lowengrub, and Shelley corrected this problem by introducing local regridding wherein vortex elements split under the action of flow deviations. Spreading and splitting provide a similar mechanism though they are dependent upon viscosity. In the uniform error

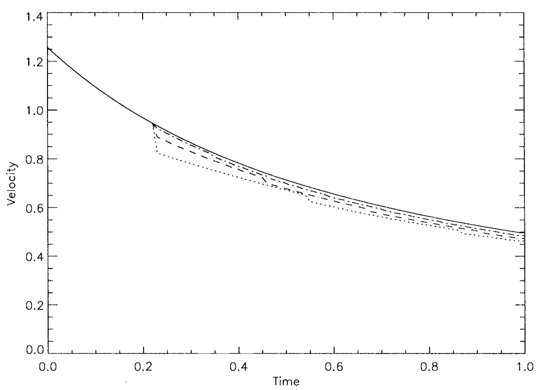


FIG. 5. *Radially symmetric patch of vorticity: velocity field at* (0,0.8) *at* $t = 1.0$ *for* $\alpha = 0.7$ (*dotted*), $\alpha = 0.8$ (*dashed*), $\alpha = 0.9$ (*dot-dashed*) *compared with the exact solution* (*solid*).

estimates of §3, the order of accuracy is actually determined to be $O(\|D\vec{u}\|l) + \ldots$ rather than just $O(l) + \ldots$. Viscosity does not enter into the error estimate because the corrected core spreading method solves the diffusion equation in the moving frame exactly. While $\vec{u}$ is not a numerical parameter it can be determined analytically and used in flow computation. This raises two issues. First, one could augment the adaptive spatial refinement process by allowing l to vary spatially with $D\vec{u}$ like $\|D\vec{u}\|l = \epsilon$ where ϵ is the new convergence parameter. Second, one could use refinement in an Euler code by allowing particles to split based on some measurement of strain during its evolution.

For this experiment, elements split more often and with greater accuracy as α grows larger. While somewhat counterintuitive, velocity measurements in Fig. 5 illustrate this increased accuracy. The refinement events can be seen as jumps in the velocity field. Refinement events begin after $t = 0.225$ because l is significantly larger than the initial core width. As $\alpha \to 1$, these jumps become more frequent but considerably smaller, demonstrating the stability of refinement.

Most importantly, a precise measurement of the uniform field error can be determined relative to the exact solution,

$$\tilde{\omega}(\vec{x}, t) = \frac{5}{4\pi(0.2^2 + 0.1t)} \exp\left(-\frac{|\vec{x}|^2}{4(0.2^2 + 0.1t)}\right). \tag{48}$$

Since l is fixed, one expects a linear profile in $1 - \alpha^2$. In Fig. 6, our results agree with this expectation. Variations at larger values of $1 - \alpha^2$ might be attributed to refinement error.

5.2. Two merging patches of vorticity. The next example is that of two blobs of the same sign placed in close proximity to one another. In the inviscid case, this experiment has been very popular because elements from opposite blobs would intertwine. In the viscous case, the blobs diffuse together at the center of the interaction, but the dynamics are still interesting. Other features of interest are the "tails" which form on the outer rim of the structure after long periods of time. We will illustrate the same concepts as in the first example, for a more challenging problem without an analytical solution. Once again, the initial conditions are simple and the algorithm will allow the problem size to expand to meet consistency requirements. Initially, there are two blobs, each with $\gamma = 5$ and $\sigma = 0.2$. In all simulations, the spatial accuracy is controlled by the parameter $l = 0.2$. (Elements larger than l split.) The elements are placed along the x-axis with a separation of 1.6, and the viscosity is 0.02. For $T = 2.5$, each blob will undergo about one quarter of a rotation about the origin, as seen in Fig. 7.

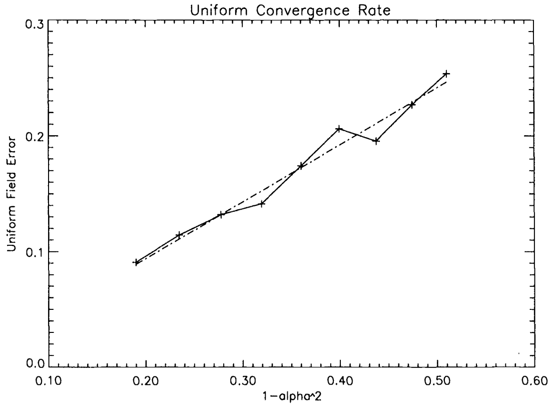


FIG. 6. *Radially symmetric patch of vorticity: uniform field error as a function of* $1 - \alpha^2$ *for* $\alpha = 0.7, 0.725, 0.75, 0.775, 0.8, 0.825, 0.85, 0.875, 0.9$ (*solid*) *and a least-squared linear fit* (*dot-dashed*).

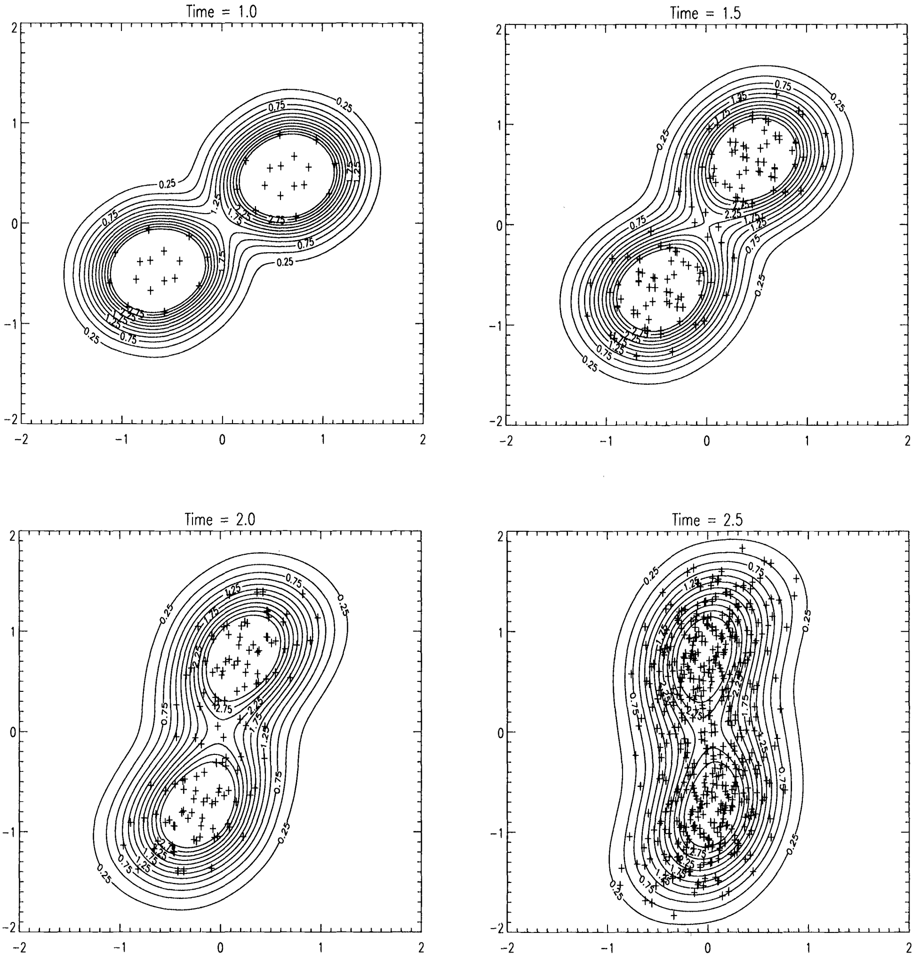


FIG. 7. *Two merging patches of vorticity: vorticity field with element positions at* $t = 1.0, 1.5, 2.0, 2.5$ *for* $\alpha = 0.8$.

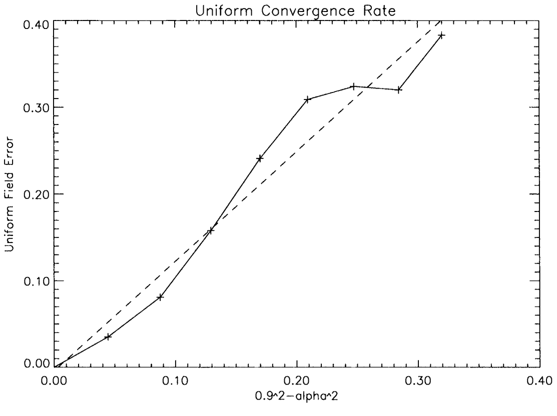


FIG. 8. *Two merging patches of vorticity: uniform field error as a function of* $(0.9)^2 - \alpha^2$ *for* $\alpha = 0.7$, 0.725, 0.75, 0.775, 0.8, 0.825, 0.85, 0.875, 0.9 (*solid*) *and a least-squared linear fit* (*dashed*).

As in the first example, the parameter α controls the relative accuracy of the method. This problem is more challenging because of the lower viscosity, greater flow deviations, and larger T. In Fig. 15 (left side only), it is clear that CCSVM naturally dedicates more elements where the vorticity is greatest. Since there is no analytic solution available, the uniform error is computed relative to the highest accuracy run at $\alpha = 0.9$. Still, the theory predicts a linear convergence in $(0.9)^2 - \alpha^2$ to this reference field and this is observed in Fig. 8.

While no analytic solution for this configuration is known, the dynamics of the second moment of this configuration,

$$\frac{d}{dt}\iint |\vec{x}|^2 \omega d\vec{x} = 4\nu \iint \omega d\vec{x}, \tag{49}$$

can be determined exactly [1]. Since vorticity is conserved, equation (49) has the exact solution

$$\iint |\vec{x}|^2 \omega d\vec{x} = 8 + 0.8t \tag{50}$$

for this particular two-vortex problem. This solution is compared with moments computed from CCSVM computations as a qualitative measurement of its accuracy. Even for the coarser value of $\alpha = 0.7$, the second moment of computed vorticity is difficult to distinguish from the exact value in Fig. 9 without adjusting the y-axis of the plot. Even so, the difference between the exact moment and the computed moment is due to the size of the time step used to advance the vortex positions rather than our method for reducing the Navier–Stokes equations to a finite system of ODEs. There is little difference between the various values of α. However, if the time step is reduced by a factor of 4, the second moments agree better even for the coarse value of $\alpha = 0.7$ (see Fig. 10).

Though this flow is relatively simple, it provides an example of how fusion can drastically improve computational efficiency. A fusion scheme using the principles in §4 will reduce the problem size during the simulation. In Fig. 3, one sees that the final problem size decreases from $N = 2048$ down to $N = 698$ with a very small loss of accuracy for $\alpha = 0.85$. Hence, fusion mediates the exponential growth caused by refinement to yield a very efficient and accurate method.

5.3. A comparative study with RWVM. To measure the strengths and weaknesses of CCSVM as a method for solving flow problems, both CCSVM and RWVM solved the previous problem on the same computer under the same conditions. These two programs are

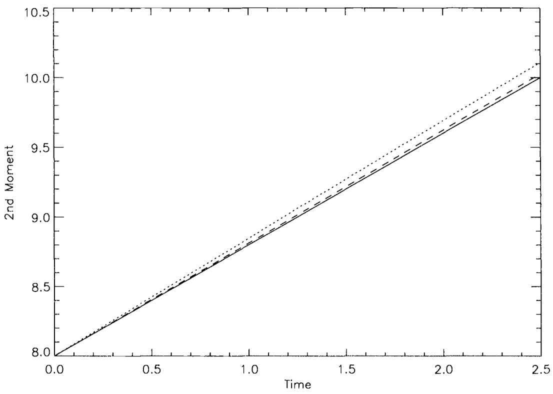


FIG. 9. *Second moment of merging pair at* $\alpha = 0.7$ (*dotted*), 0.8 (*dashed*), *and* 0.9 (*dot-dashed*) *with the exact value* (*solid*).

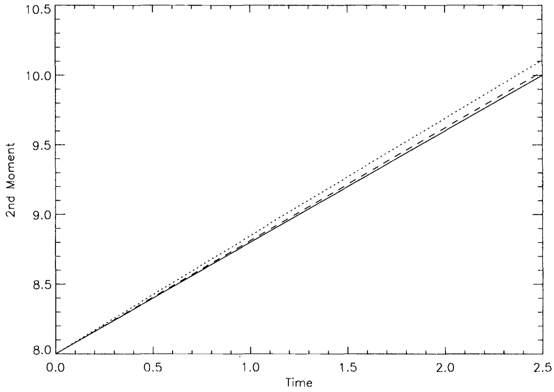


FIG. 10. *Second moment of merging pair at* $\alpha = 0.7$ *and* $\Delta t = 0.005$ (*dotted*) *and* 0.00125 (*dashed*) *with the exact value* (*solid*).

identical except for subroutines handling the diffusion. For CCSVM, elements spread and, if necessary, split. For RWVM, elements undergo random walks. Both programs were compiled with the same compiler settings on a Sparc 2 workstation without any special vectorization or parallelization. (Since both algorithms are fully parallelizable and vectorizable, specialized hardware is not likely to affect the outcome anyway.) Though both methods would have benefited from a fusion algorithm, this might have introduced some uncertainty into the experiment because fusion is geometry- and index-dependent. Thus, neither program used a fusion algorithm. Under these conditions, it was possible to directly compare these two algorithms.

The initial conditions for CCSVM were identical to those of the previous problem. For RWVM, $\frac{N}{2}$ vortex elements of width $l = 0.2$ and circulation $\frac{10}{N}$ were placed at $(0.8, 0)$, and the other $\frac{N}{2}$, having the same circulation and width, were placed at $(-0.8, 0)$. Thus, both methods began with the exact same initial conditions. Since the core sizes were the same, both methods had the same spatial accuracy during the convective step. Indeed, since both methods differed only in how they approximate the diffusive term, it was necessary to hold the core size constant to measure any differences in the methods.

Since the problem size for CCSVM changes with time and the time complexity is $O(N^2)$, a random walk algorithm with size $N = N_0$ should require the same amount of time as CCSVM with α chosen so that $\sqrt{\langle N^2 \rangle} = N_0$, where $\langle N^2 \rangle$ denotes the ensemble average of

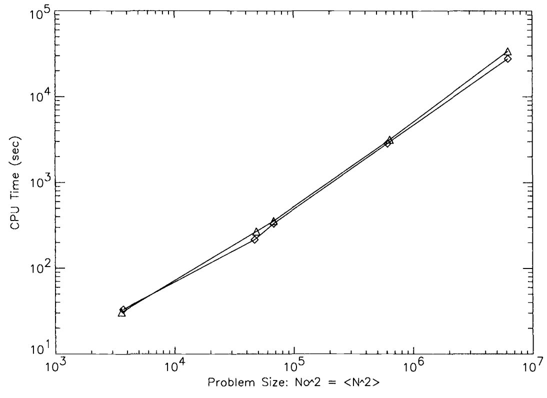


FIG. 11. *Two merging patches of vorticity: comparison of CPU time expended vs. square of problem size for RWVM (triangle) and CCSVM (diamond).*

N^2 over time. As predicted, both methods used about the same amount of computer time for a given problem size. Also, the relationship between N_0^2 and CPU time in Fig. 11 is linear for both methods as expected. If anything, the core spreading methods ran slightly faster, but one can safely say that both methods require roughly the same amount of computational effort.

Both methods conserve the zeroth moment, but only CCSVM conserves the first moment exactly. This is especially evident with smaller problem sizes as in Fig. 12. Using random walks when the problem size is low can result in very unphysical behavior. Even in larger simulations, regions of the flow that are approximated by a low particle density would be strongly affected by statistical noise [21]. Though both methods suffer from time integration errors when measuring their second moment, the random walk method appears to suffer more. Though both methods are convergent, CCSVM converges while conserving these important physical properties which are built into the algorithm. Figure 13 displays first and second moments for relatively large problems sizes.

The final comparison involves measuring the performance of both methods in terms of uniform error as a function of problem size. To eliminate any CCSVM bias, a random walk simulation using a large number of particles ($N = 5000$) was run on a Convex C-120 mini-supercomputer using a small step size ($\Delta t = 0.00125$) and treated as an exact solution for comparison with other data sets. The results, shown in Fig. 14, indicate that CCSVM has slightly better convergence properties than RWVM as measured by the L^∞ norm. For both methods, the accuracy as a function of problem size is similar to that observed by Sethian in his study of random walk vortex method performance while simulating boundary layer instability [17]. Moreover, using the random walk solution, one obtains the predicted $1 - \alpha^2$ linear rate of convergence. Even though both methods appear to indicate the same rate of convergence, CCSVM is more accurate for a given problem size and so produces better results given a fixed amount of computational resources, as can be seen in Fig. 15.

6. Conclusion. This paper introduces and analyzes a new vortex method with many attractive properties. Since both spreading and adaptive spatial refinement are local processes, this method is well suited for vector and parallel computation. These theoretical findings are sufficient to assure application-oriented investigators that certain uniform convergence properties will be achieved. The introduction of the numerical parameter α overshadows the importance of l for viscous computations once l is chosen to resolve given flow features. Finally, the exponential growth in the problem size can be controlled by the fusion process

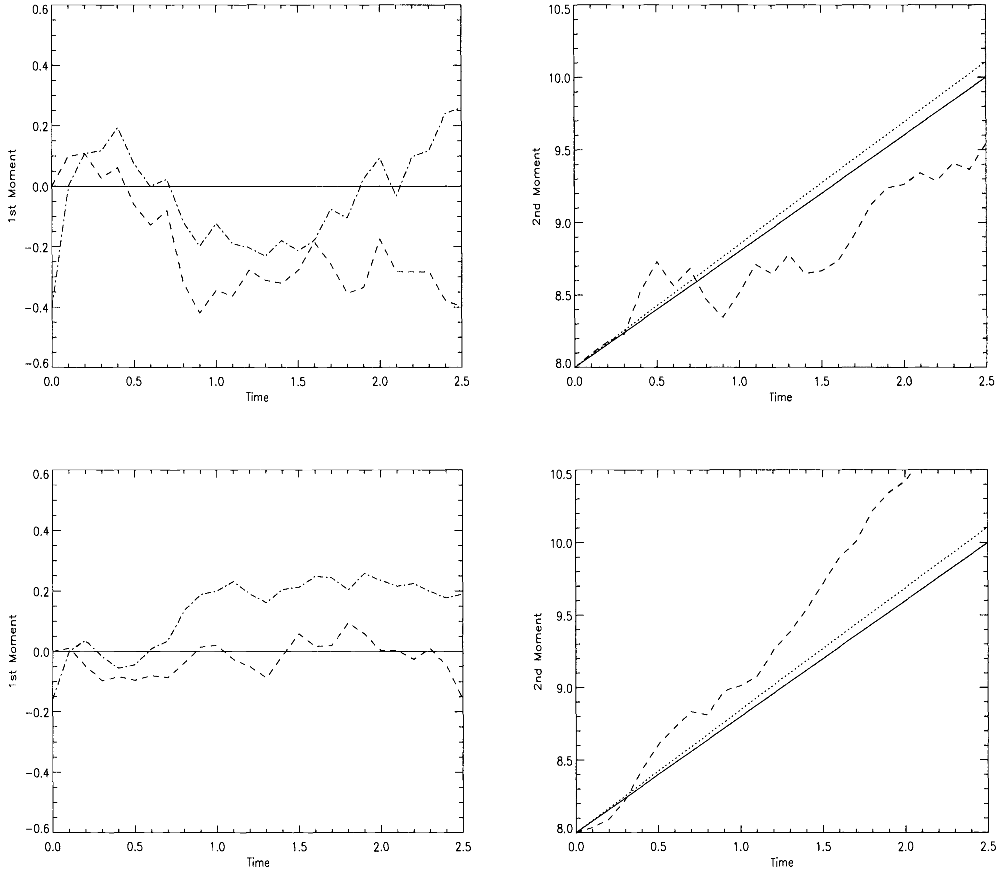


FIG. 12. *Measurements of moments comparing CCSVM with RWVM. Top row: CCSVM with* $\sqrt{\langle N^2\rangle} = 60.9$ ($\alpha = 0.7$) *and RWVM with* $N = 60$. *Bottom row: CCSVM with* $\sqrt{\langle N^2\rangle} = 216.4$ ($\alpha = 0.8$) *and RWVM with* $N = 220$. *Left: first x (solid) and y (dotted covered by solid) moment of CCSVM (solid covering dotted) and first x (dashed) and y (dot-dashed) of RWVM. Right: second moment of exact solution (solid), CCSVM (dotted), and RWVM (dashed).*

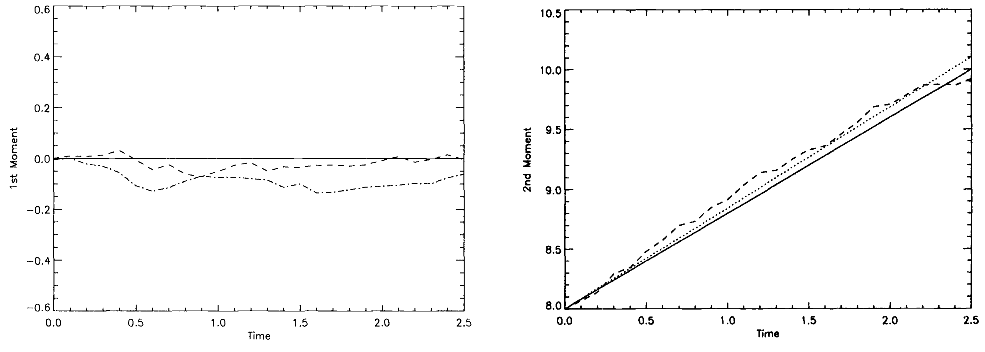


FIG. 13. *Measurements of moments comparing CCSVM with* $\sqrt{\langle N^2\rangle} = 782.8$ ($\alpha = 0.85$) *and RWVM with* $N = 800$. *Left: first x (solid) and y (dotted covered by solid) moment of CCSVM (solid covering dotted) and first x (dashed) and y (dot-dashed) of RWVM. Right: second moment of exact solution (solid), CCSVM (dotted), and RWVM (dashed).*

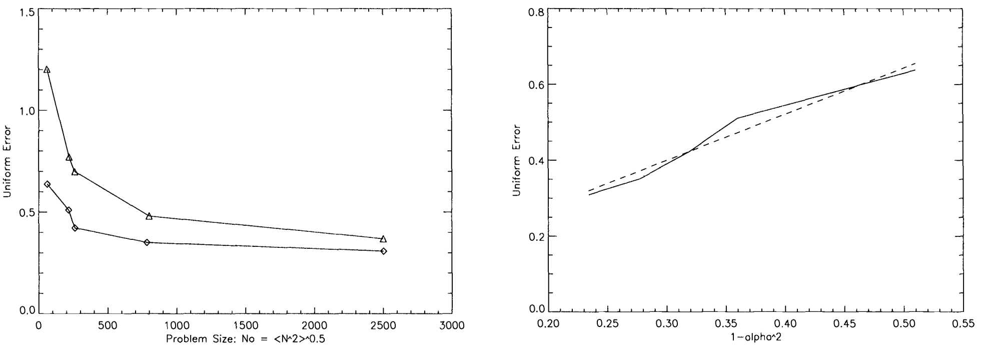


FIG. 14. *Measurements of the uniform field error of CCSVM. Left: uniform error as a function of problem size for RWVM (triangle) and CCSVM (diamond). Right: uniform field error measured from high resolution random walk solution as a function of* $1 - \alpha^2$ *for* $\alpha = 0.7, 0.8, 0.825, 0.85, 0.875$ *(solid) and a least-squared linear fit (dashed).*

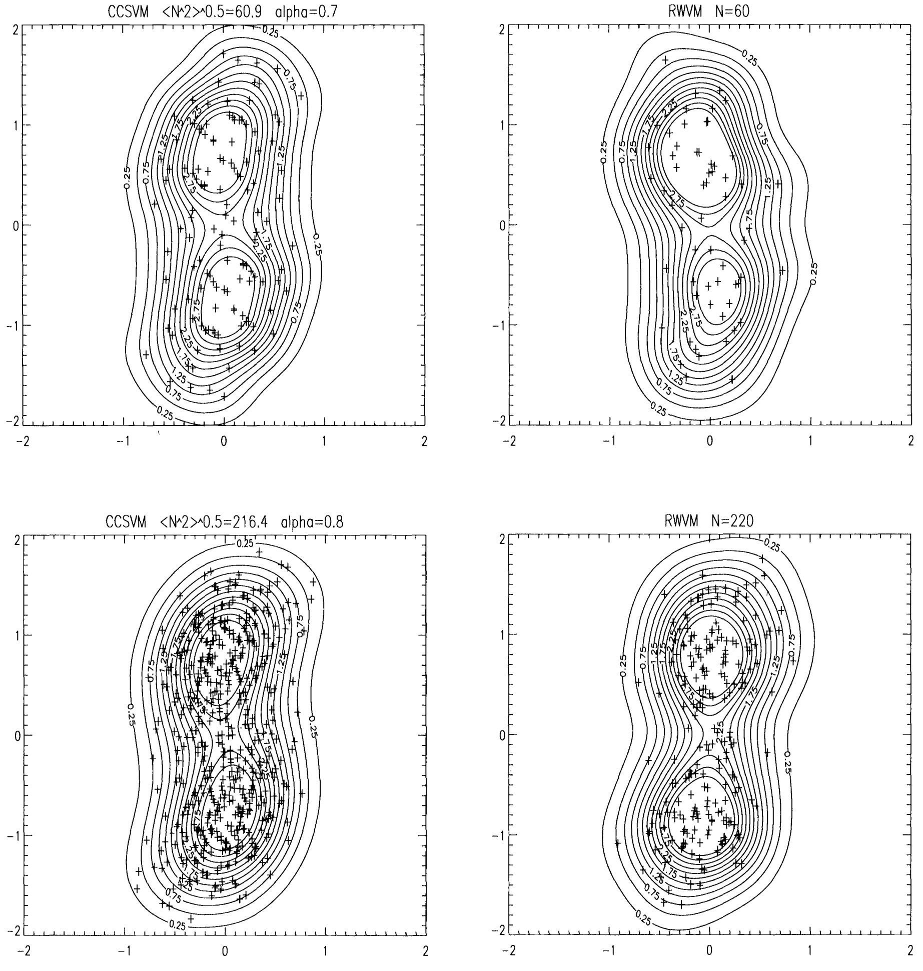


FIG. 15. *Comparing CCSVM and RWVM simulations for comparable problem sizes. Vorticity field with element positions at* $t = 2.5$ *for a small problem size (top) and a moderate problem size (bottom). Plots on the left correspond to CCSVM while plots on the right correspond to RWVM.*

analyzed in this paper. We demonstrated the accuracy and effectiveness of this method with nontrivial examples and found the results to be reassuring. Among these are measurements of the uniform error in the vorticity field in agreement with the derived theory.

While this paper attempts to analyze some important questions about spreading blob methods, there are still many unexplored areas to study. Only one stable refinement process is discussed in this paper, but many others are possible. Perhaps there is an optimal one in some sense. In proving linear convergence for CCSVM in L^∞, a full nonlinear uniform convergence proof introduces itself as the next challenge to investigators.

Nonetheless, linear convergence is still a useful result. Since uniform convergence of the vorticity field implies uniform convergence of the velocity field, one could view the full nonlinear residual operator as the linear theory R with a small perturbation as numerical parameters approach their limiting values.

Also, Condition A and Condition B can be applied to the convergence of other grid-free and gridded schemes. While these conditions were applied specifically to CCSVM, the general framework can be applied to any numerical method. Together, they form the basis for an alternative convergence theory.

While many investigators have methods for removing extraneous vortex elements from a simulation, fusion is a general, well-defined method for controlling the problem size. Maintaining efficient problem sizes during a vortex simulation is crucial because the natural action of viscosity is to increase the problem regardless of the method used. Furthermore, if one extends this work to bounded flows, vorticity will be shed from boundaries into the flow to satisfy the no-slip condition, increasing the problem size even more. Fusion is a natural way to recombine redundant elements while conserving several moments. Even without viscosity, Hou, Lowengrub, and Shelley have demonstrated the importance of splitting vortex elements to prevent the method from losing accuracy in finite time. As mentioned earlier in §5, it may be possible to adapt refinement to inviscid flows by computing flow deviations.

Acknowledgments. I would like to thank my advisor, Bruce Bayly, for suggesting this problem and giving me the freedom to pursue it. Also, I would like to acknowledge his friendship, advice, encouragement, and support. Furthermore, I revised this paper with the assistance of English Professor Thomas Miller during his fantastic Advanced Technical Writing Seminar, a summer course which proved to be an invaluable resource. The majority of the work in this paper was done at the University of Arizona's Program in Applied Mathematics.

REFERENCES

[1] G. K. BATCHELOR, *An Introduction to Fluid Dynamics*, Cambridge University Press, 1967, pp. 528-530.

[2] J. T. BEALE AND A. MAJDA, *Vortex methods.* II: *Higher order accuracy in two and three dimensions*, Math. Comp., 39 (1982), pp. 29-52.

[3] A. J. CHORIN, *Numerical study of slightly viscous flow*, J. Fluid Mech., 57 (1973), pp. 785-796.

[4] ———, *Vortex sheet approximation of boundary layers*, J. Comp. Phys., 27 (1978), pp. 428-442.

[5] ———, *Vortex models and boundary layer instability*, SIAM J. Sci. Stat. Comput., 1 (1980), pp. 1-21.

[6] W. FELLER, *An Introduction to Probability Theory and Its Applications*, Chap. 7, John Wiley, New York, 1957, pp. 168-173.

[7] J. GOODMAN AND T. Y. HOU, *New stability estimates for the 2-D vortex method*, Comm. Pure Appl. Math., 44 (1991), pp. 1015-1031.

[8] C. GREENGARD, *The core spreading vortex method approximates the wrong equation*, J. Comp. Phys., 61 (1985), pp. 345-348.

[9] O. H. HALD, *Convergence of vortex methods for Euler's equations.* II, SIAM J. Numer. Anal., 16 (1979), pp. 726-755.

[10] ———, *Convergence of vortex methods for Euler's Equations,* III, SIAM J. Numer. Anal., 24 (1987), pp. 538-582.

[11] T. Y. Hou, J. Lowengrub, and M. J. Shelley, *Exact desingularization and local regridding for vortex methods*, in Lectures in Applied Mathematics: Vortex Methods and Vortex Dynamics, C. Greengard, ed., American Mathematical Society, Providence, RI, 1991, pp. 341-362.

[12] E. Isaacson and H. B. Keller, *Analysis of Numerical Methods*, John Wiley, New York, 1966, pp. 514-529.

[13] Z. Y. Lu and T. J. Ross, *Diffusing-vortex numerical scheme for solving incompressible Navier-Stokes equations*, J. Comp. Phys., 95 (1991), pp. 400-435.

[14] S. Mas-Gallic, *Deterministic particle methods: Diffusion and boundary conditions*, in Lectures in Applied Mathematics: Vortex Methods and Vortex Dynamics, C. Greengarde, ed., American Mathematical Society, 1991, pp. 433-465.

[15] L. F. Rossi, *A Spreading Blob Vortex Method for Viscous Bounded Flows*, Ph.D. thesis, University of Arizona, Tuscon, AZ, December 1993.

[16] ———, *Merging computational elements in Lagrangian simulations*, SIAM J. Sci. Comput., submitted, 1995.

[17] J. A. Sethian, *On measuring the accuracy of the vortex method: Using a random method to model stable and unstable flow*, in Lecture Notes in Mathematics 1360: Vortex Methods, C. Anderson and C. Greengard, eds., Springer-Verlag, Berlin, 1987, pp. 83-93.

[18] ———, *A brief overview of vortex methods*, in Vortex Methods and Vortex Motion, K. Gustafson and J. A. Sethian, eds., Frontiers in Applied Mathematics, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1990.

[19] J. A. Sethian, J.-P. Brunet, A. Greenberg, and J. P. Mesirov, *Two-dimensional, viscous, incompressible flow in complex geometries on a massively parallel processor*, J. Comp. Phys., 101 (1992), pp. 185-206.

[20] A. Tucker, *Applied Combinatorics*, Chap. 5.5, John Wiley, 2nd ed., 1984, pp. 204-207.

[21] L. Van Dommelen, *A Vortex Redistribution Technique*, Tech. Rep. FMRL TR-3, Florida State University, Tallahassee, FL, August 1989.

SIAM J. SCI. COMPUT.
Vol. 17, No. 2, pp. 398–415, March 1996

006

FAST FOURIER TRANSFORM ACCELERATED FAST MULTIPOLE ALGORITHM*

WILLIAM D. ELLIOTT† AND JOHN A. BOARD, JR.†

Abstract. This paper describes an $\mathcal{O}(p^2 \log_2(p)N)$ implementation of the fast multipole algorithm (FMA) for N-body simulations. This method of computing the FMA is faster than the original, which is $\mathcal{O}(p^4N)$, where p is the number of terms retained in the truncated multipole expansion representation of the potential field of a collection of charged particles. The p term determines the accuracy of the calculation. The limiting $\mathcal{O}(p^4)$ computation in the original FMA is a convolution-like operation on a matrix of multipole coefficients. This paper describes the implementation details of a conversion of this limiting computation to linear convolution, which is then computed in the Fourier domain using the fast Fourier transform (FFT), based on a method originally outlined by Greengard and Rokhlin. In addition, this paper describes a new block decomposition of the multipole expansion data that provides numerical stability and efficient computation. The resulting $\mathcal{O}(p^2 \log_2(p))$ subroutine has a speedup of 2 on a sequential processor over the original method for $p = 8$, and a speedup of 4 for $p = 16$. The new subroutine vectorizes well and has a speedup of 3 on a vector processor at $p = 8$ and a speedup of 6 at $p = 16$.

Key words. N-body problem, many-body problem, fast multipole algorithm, fast multipole method, tree codes, molecular dynamics, fast Fourier transform

AMS subject classifications. 70-08, 70F10, 42A85, 42C10, 82C22, 33C55, 65C20

1. Introduction. The Greengard–Rokhlin fast multipole algorithm (FMA) [6, 9] is an efficient and inherently parallel algorithm for computing the Coulomb (electrostatic) or Newtonian (gravitational) potential and forces in many-particle systems. The FMA is an $\mathcal{O}(N)$ solution to the N-body problem, which is a significant improvement in performance over the simpler but costly $\mathcal{O}(N^2)$ method of directly computing the potentials and forces between all pairwise combinations of particles. Applications of the FMA include simulations of molecular dynamics, astrophysical processes, and fluid dynamics, and many parallel implementations have been reported [1, 3, 4, 5, 7, 8, 13, 15, 16].

This paper describes an improvement to the FMA that applies to implementations on sequential or parallel machines. The improvement is a speedup of the limiting subroutine in the FMA by converting the computation to the Fourier domain, as described originally by Greengard and Rokhlin [10]. The speedup also applies to related algorithms, such as that of Barnes and Hut [2], when these are extended to include many terms in the multipole expansions.

Section 2 of the paper gives a brief overview of the FMA and describes the limiting multipole computation. Section 3 describes converting the limiting computation to the Fourier domain. To ensure a numerically stable and efficient computation, several optimizations are required on the basic Fourier domain computation; these improvements are described in §4. Section 5 reports results of running the FFT algorithm on a RISC processor and on a vector supercomputer.

2. Overview of the FMA. The FMA is described in detail in [6]. We use the language of electrostatics for our development, because electrostatic applications (such as molecular dynamics simulations) particularly benefit from the work herein (see §6). The algorithm begins with a hierarchical spatial decomposition of the computational cube that contains the particles

*Received by the editors March 7, 1994; accepted for publication (in revised form) October 28, 1994. Work was supported in part by NSF Engineering Research Center grant CDR-8622201, NSF Grand Challenge Applications Group award ASC-9318159, NIH Research Resource award RR08102-01, and a grant of computer time from the North Carolina Supercomputing Center.

†Duke University, Department of Electrical Engineering, P.O. Box 90291, Durham, NC 27706-0291 (welliott@ee.duke.edu, jab@ee.duke.edu).

of interest into successively smaller cells. Each cell is split into eight child cells, allowing a hierarchical oct-tree representation of the simulation region. With a uniform distribution of charged particles, a uniform oct-tree is produced; the uniform case is assumed for this paper. Adaptive variants of the FMA exist for nonuniform particle distributions [5], and the FFT methods developed here apply equally to that case.

To improve the efficiency of the potential and force calculations, the FMA replaces most of the particle-to-particle interactions with cell-to-cell interactions by first representing the aggregate potential field of a group of charges in a cell with an equivalent multipole expansion. Interactions between cells are then computed by manipulating the coefficients of multipole expansions.

Multipole expansions corresponding to the cells at the lowest level of spatial decomposition (the leaf nodes of the oct-tree) are computed via (1). For a given leaf cell containing k particles, the position of the ith particle is given in spherical coordinates by $(\rho_i, \alpha_i, \beta_i)$. $\Phi(\mathbf{r})$ represents the aggregate potential field due to all k particles at any point $\mathbf{r} = (r, \theta, \phi)$ outside the given cell.The infinite multipole expansion describes the potential field exactly. The series is summed over the defined range of the order l $(0 \leq l \leq \infty)$ and degree m $(-l \leq m \leq l)$ of the spherical harmonic function $Y_l^m(\theta, \phi)$.[1] Outside this allowed range of values for l and m, Y_l^m is taken to be zero.

$$\Phi(\mathbf{r}) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \frac{M_l^m}{r^{l+1}} Y_l^m(\theta, \phi), \tag{1}$$

$$M_l^m = \sum_{i=1}^{k} q_i \rho_i^l Y_l^{*m}(\alpha_i, \beta_i). \tag{2}$$

The FMA next computes multipole expansions for all cells higher up the tree at all levels by translating the origins of the multipole expansions of the eight child cells of each parent cell to the center of the parent cell using a multipole-to-multipole (M2M) translation in an upward pass through the oct-tree.

We say that two cells A and B from the same or adjacent levels of the tree are *well separated* if they are sufficiently far apart so that the multipole series of A converges fairly rapidly at the center of B. Pragmatically, A and B are separated by at least two cell widths. Once all multipole expansions are computed at all levels, well-separated cells interact by multipole-to-local (M2L) translation. In M2L, a local Taylor series expansion of the cell A potential is formed, centered on cell B, to evaluate the potential at the particle sites in B due to all the particles in A. As B interacts with other well-separated cells, the results are accumulated into a single local (Taylor) series. During the downward pass of the algorithm, parents pass the Taylor series representing their far-field interactions to each of their children using local-to-local (L2L) translation; the series is recentered in each child box.

Although computing the cell-to-cell interactions is a complicated procedure compared with the simpler particle-to-particle $\frac{1}{r}$ computation, for a simulation with a large enough number of particles, the multipole-based cell-to-cell interactions will run faster. For details on the FMA, the reader is directed to [1, 3, 5, 7, 8, 13, 15, 16].

This paper will focus on the mathematics of the three cell-to-cell translations (M2M, M2L, and L2L), depicted in Figure 1. In particular, the M2L interaction between well-separated cells will be discussed because it is the performance limiting subroutine in the FMA computations.

[1] Consistent with Greengard and Rokhlin's notation [6], the spherical harmonic functions Y_l^m in this paper omit the standard normalization factor $\sqrt{(2l+1)/4\pi}$.

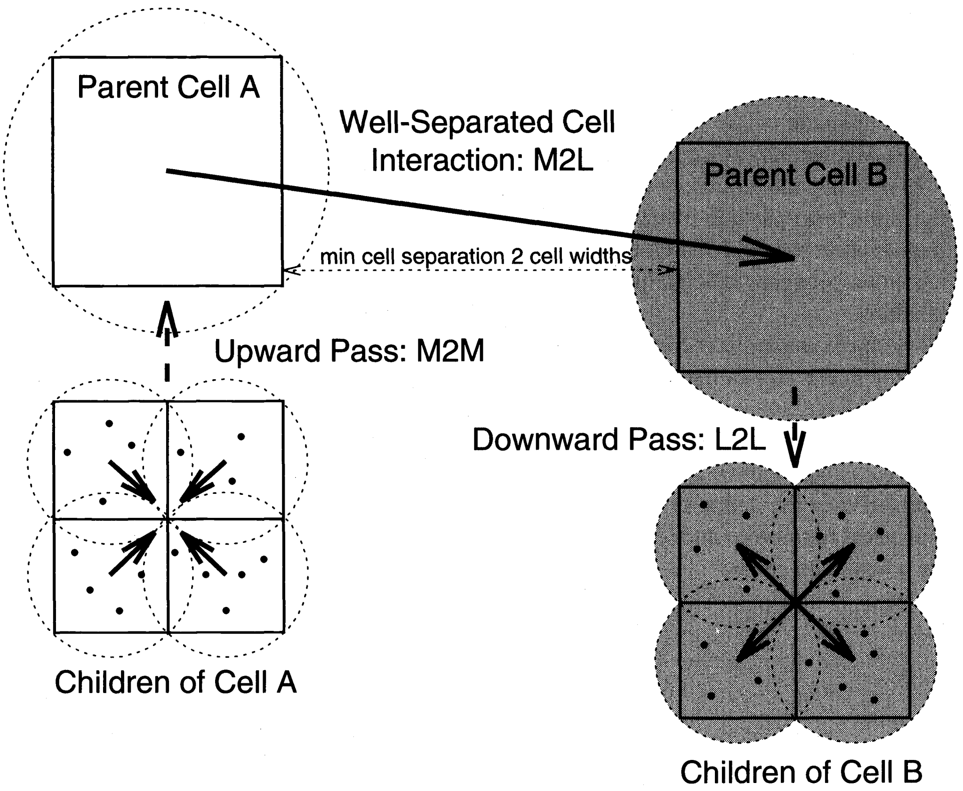


FIG. 1. *FMA multipole expansion translations. The children of cell A use the* M2M *translation to shift to the center of cell A the multipole expansions representing the far-field potential of the particles contained in them. The* M2L *translation allows well-separated cells to interact by creating a local (Taylor) series expansion of the far-field potential of distant cell A around the center of cell B. Using the* L2L *translation, cell B in turn shifts the center of the local expansion to the center of each of its children.*

2.1. Contribution to the FMA runtime of the M2L subroutine. In the FMA, the M2L translation, which is the interaction between well-separated cells, consumes the most runtime. The M2L subroutine dominates the runtime due to the number of cells in the oct-tree and the number of distant cells with which each cell must interact. If level 0 is the original simulation space, each level L of spatial decomposition creates 8^L new cells. The interaction list of each cell i contains all well-separated cells that have not already appeared in the interaction lists of i's parent. (The exact rule for assigning interaction lists is somewhat more complex; see [6].) Using a well-separated criteria of two cell widths puts as many as 189 peer-level and parent-level cells in the interaction list of any cell [16]. Figure 2 shows the total number of M2L interactions (or equivalently, calls to the M2L subroutine) at each level of spatial decomposition in the FMA. In contrast to the "horizontal" M2L interactions, the M2M and L2L translations are "vertical" between parent and children only, and thus are performed eight times at most for each cell. Although the M2L, M2M, and L2L routines have a similar runtime cost per call, the large number of M2L interactions causes that subroutine to dominate the FMA runtime.

Multipole expansions and the three translation operations acting on them in the FMA are exact in the limit of an infinite series. When truncated to a finite number of terms p, the relative error in the potential is bounded above by $(\frac{1}{2})^p$ as Greengard and Rokhlin proved [6].

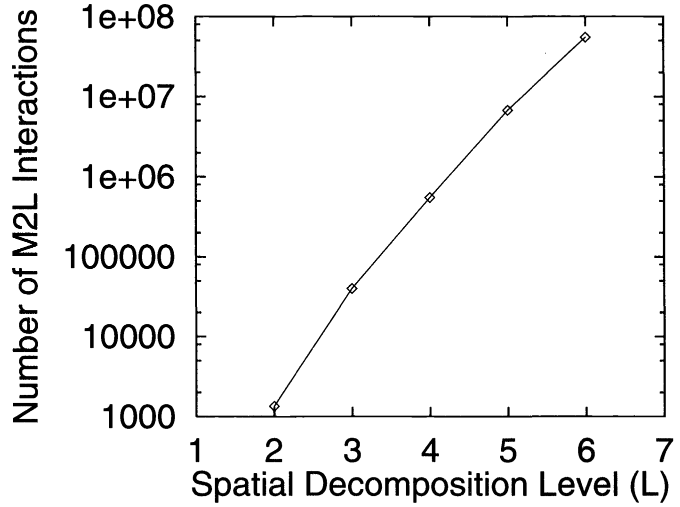


FIG. 2. M2L *interactions in the FMA. The plot shows the rapid growth in the number of* M2L *interactions, measured by the number of calls to the* M2L *subroutine, as the level of spatial decomposition increases to accommodate more particles.*

In practice, p is the linear dimension of a 2-D matrix of multipole expansion coefficients; there are a total of p^2 terms in the matrix. Each interaction between a pair of cells in the FMA requires the translation of the origin of a series representation of the electrostatic field; this in turn requires a double summation over all terms of the input multipole expansion coefficient matrix multiplied with a corresponding term in a weighting function matrix to produce each term in the output coefficient matrix (see (4) below). These operations yield $\mathcal{O}(p^4)$ complexity for the series translations.[2] These translations resemble 2-D discrete convolution, which is also $\mathcal{O}(p^4)$. The complexity of the operations on the multipole expansions, combined with the requirement to perform a large number of M2L translations for each cell, causes that part of the FMA to dominate the runtime over the accuracy range (values of p) of interest for molecular dynamics.

3. Conversion to Fourier domain. The convolution-like form of the computation and the expense of the M2L translation motivate the conversion of the operation to the Fourier domain using a 2-D FFT. In a technical report, Greengard and Rokhlin [10] described an FFT conversion of the FMA. Schmidt and Lee [15] anticipated an FFT conversion of the FMA as an optimization. Pan, Reif, and Tate [14] described a theoretical parallelized algorithm based on Greengard and Rokhlin's FFT conversion that could further improve the runtime of such an algorithm when parallelized. Here we describe the implementation considerations and performance of the FFT version of the FMA implemented in an existing version of FMA code developed by Board and Leathrum [4, 12]. To our knowledge this is the first published version of the implementation details of an FFT version of the FMA.

3.1. Description of the M2L translation. The M2L translation from a distant cell A to a local cell B, see (3), simultaneously translates the origin O of a truncated multipole expansion describing the potential field due to the particles in A to a new origin at point $O' = (\rho, \alpha, \beta)$ at the center of B and converts that expansion to a local Taylor series expansion about the new

[2] Greengard and Rokhlin have reported a method that takes advantage of symmetry to reduce many of the translations to $\mathcal{O}(p^2)$ [11].

origin:

$$\Phi(\mathbf{r}) = \Phi'(\mathbf{r}') = \sum_{l=0}^{p-1} \sum_{m=-l}^{l} \mathbf{M}_l^m \frac{Y_l^m(\theta, \phi)}{r^{l+1}} = \sum_{l'=0}^{p-1} \sum_{m'=-l'}^{l'} \mathbf{L}_{l'}^{m'} Y_{l'}^{m'}(\theta', \phi') r'^{l'}. \tag{3}$$

Within the error bound described in §2.1, the local expansion of the potential field at any arbitrary point (r', θ', ϕ') with respect to the new origin O' gives the same value for the potential due to A as the multipole expansion evaluated at the same point (r, θ, ϕ) described with respect to the old origin O, provided series convergence criteria are met. Cells from the same level of the tree separated by at least two cell-widths meet the series convergence criteria. Since the location of the evaluation point of the potential is arbitrary as far as the M2L translation is concerned, the translation can be viewed as an operation on only the coefficients of the input matrix of multipole expansion coefficients $\mathbf{M}_l^m$, using a matrix of translation coefficients $\mathbf{T}_{l,m,l',m'}^{M2L}$ as a weighting function, resulting in an output matrix of local expansion coefficients $\mathbf{L}_l^m$ as shown in (4) (using the notation of Schmidt and Lee [15]):

$$\mathbf{L}_{l'}^{m'} = \sum_{l,m} \mathbf{T}_{l,m,l',m'}^{M2L} \mathbf{M}_l^m, \tag{4}$$

$$\mathbf{T}_{l,m,l',m'}^{M2L} = \frac{(-1)^{l'+m} A_l^m A_{l'}^{m'}}{A_{l'+l}^{m'-m}} \frac{Y_{l'+l}^{*m'-m}(\alpha, \beta)}{\rho^{l'+l+1}}, \tag{5}$$

$$A_l^m = \frac{(-1)^{l+m}}{\sqrt{(l+m)!(l-m)!}}. \tag{6}$$

These input, output, and translation coefficient matrices are approximately half-filled, since the coefficients are zero outside the allowed range of l and m, where l is a nonnegative integer and $-l <= m <= l$. The spherical harmonic function $Y_l^m(\theta, \phi)$ equals zero for index values outside of this range.

3.2. Converting M2L to convolution form. Double summation over an input coefficient matrix multiplied by a weighting matrix resembles convolution. In fact the M2L computation resembles linear convolution with what can be considered a 4-D weighting function matrix $\mathbf{T}_{l,m,l',m'}^{M2L}$; the four dimensions are l, m, l', and m'. In order to reduce the complexity, weighting coefficients are needed that are not a function of all four indices but are a function only of the difference between the corresponding indices $l' - l$ and $m' - m$. In that case, the computation would be a convolution of the 2-D input coefficient matrix with a 2-D transfer function matrix we will call $\mathbf{H}_l^m$.

3.3. Warping the coefficients with the A_l^m function. Equations (7) and (8) show the 2-D transfer function $\mathbf{H}_l^m$ created by moving to the left side of (4) the terms of $\mathbf{T}_{l,m,l',m'}^{M2L}$ that depend on only l' and m' and grouping with $\mathbf{M}_l^m$ the terms that depend on only l and m. The result "warps" the input coefficient matrix with the A_l^m normalization function and produces a warped output coefficient matrix.[3] Of course, the output values must be unwarped with the A_l^m function before computing any potentials, but by using warped values, the double summation of input coefficients multiplied by transfer function coefficients is a 2-D linear convolution operation. We have

$$\left[(-1)^{-l'} \frac{\mathbf{L}_{-l'}^{m'}}{A_{-l'}^{m'}}\right] = \sum_{l,m} \left[\mathbf{H}_{-(l'-l)}^{m'-m}\right] \left[(-1)^m \mathbf{M}_l^m A_l^m\right], \tag{7}$$

[3] The A_l^m function used here differs slightly from that of Greengard and Rokhlin [6, 9].

$$[\mathbf{H}_l^m] = \frac{Y_l^{*m}(\alpha, \beta)}{A_l^m \rho^{l+1}}. \tag{8}$$

The equation in this form can be computed more quickly in the Fourier domain using the FFT of the respective matrices, as shown in (9)-(13). Defining the "warped" functions $y(l, m)$, $h(l, m)$, and $x(l, m)$, we perform the convolution computation $h \circledast x$ in the Fourier domain by element-by-element multiplication of the matrices $H(\omega_{-l}, \omega_m)$ and $X(\omega_l, \omega_m)$, resulting in the output matrix Fourier space matrix $Y(\omega_{-l}, \omega_m)$. The 2-D FFT of the coefficient matrices has a complexity $\mathcal{O}(p^2 \log_2(p))$, and the element-by-element multiplication of matrix elements has a complexity $\mathcal{O}(p^2)$. We have

$$y(l, m) = \left[(-1)^l \frac{\mathbf{L}_l^m}{A_l^m}\right], \tag{9}$$

$$h(l, m) = [\mathbf{H}_l^m] = \frac{Y_l^{*m}(\alpha, \beta)}{A_l^m \rho^{l+1}}, \tag{10}$$

$$x(l, m) = [(-1)^m \mathbf{M}_l^m A_l^m], \tag{11}$$

$$y(-l', m') = \sum_{l,m} h(-(l' - l), m' - m) x(l, m) = h(-l', m') \circledast x(l', m'). \tag{12}$$

With the equation in convolution form, the distinction between primed and unprimed values of l and m disappears, and the M2L translation can be done in the Fourier domain. We have

$$y(-l, m) = h(-l, m) \circledast x(l, m) \leftrightarrow Y(\omega_{-l}, \omega_m) = H(\omega_{-l}, \omega_m) X(\omega_l, \omega_m). \tag{13}$$

The $-l$ terms in the transfer function H and the output Y affect the alignment of those matrices in coefficient space. Specifically, the transfer function matrix and the output matrix both will be aligned with the input matrix in the m direction but reversed in the l direction. The FFT versions of the M2M and L2L translations, shown in the Appendix, are similar to the M2L equations.

4. Implementation considerations and enhancements. While the mathematics of §3 appear complete, there are several modifications required for actual computation of the FFT version of the M2L to achieve correctness and numerical stability. Also, to run efficiently enough to outperform the conventional method of calculating the M2L interaction at low p values, several optimizations are necessary. This section describes the modifications necessary for a correct and efficient implementation.

4.1. Linear convolution from FFT circular convolution. Computing the M2L translation in Fourier space using (13) introduces two side effects that do not occur in the original coefficient-space computation (3). First, to get the required linear convolution from FFT circular convolution requires zero padding, which increases the complexity of the computation. Nominally, zero padding will result in an FFT matrix of size $4p \times 2p$. $4p$ places are required in the m direction because the coefficients range from $m = -(p - 1)$ to $(p - 1)$ (including 0) at the widest point and from $l = 0$ to $(p - 1)$ at the highest point. The second side effect occurs when FFT convolution produces nonzero coefficients outside the allowed range of l and m ($l \geq 0$ and $|m| \leq l$); while the FFT method computes values over the full $4p \times 2p$ matrix, the conventional coefficient-space computation of the M2L translation needs to sum only over the allow ranges of l, m, l', and m'.

One problem associated with these side effects is the increase in runtime complexity of the computation. To illustrate, the FFT of an $N \times M$ matrix requires $\frac{NM}{2}(\log_2 N + \log_2 M)$ multiplications of complex numbers. For the zero padded $4p \times 2p$ matrix, then, the number of complex multiplications is $4p^2(2\log_2 p + 3)$ for each matrix. After the FFT, the element-by-element matrix multiplication requires $8p^2$ complex multiplications. In contrast, it can be shown that the original coefficient-space M2L computation requires $\frac{p^4+4p^3+5p^2+2p}{12}$ complex multiplications. The problem arises because typical values of p are low ($p \leq 16$), and the coefficient-space computations for those p values are less than the equivalent FFT computations using the full $4p \times 2p$ matrix.

Fortunately, half of the zero padding can be eliminated as follows. The wrap-around effect of FFT circular convolution in the m direction of an array $2p$ wide affects only those values that are outside of the allowed range of l and m anyway, allowing the FFT matrix to shrink down to $2p \times 2p$, which eliminates half of the zero padding in the m direction and still gives correct results. Furthermore, the 1-D FFT in the m direction contains significant redundancies due to the following relationships among the values of the coefficients with respect to m:

$$(14) \qquad M_l^{-m} = (-1)^m M_l^{*m}, \quad L_l^{-m} = (-1)^m L_l^{*m}.$$

This relationship applies to the multipole expansion coefficients and to the local expansion coefficients as well as the spherical harmonic function:

$$(15) \qquad Y_l^{-m}(\theta, \phi) = (-1)^m Y_l^{*m}(\theta, \phi).$$

When converted to the Fourier domain, this relationship results in half of the 1-D FFT coefficients in the m direction being equal to the complex conjugates of the other half. Significantly, this relationship holds even for the $2p \times 2p$ coefficient matrix described above subject to the wrap-around effect of circular convolution. Thus, one half of the FFT array in the m direction contains all of the information about the multipole expansion or local expansion coefficients. Therefore, only half the array need be computed and stored, saving memory and runtime. The values of the unused half of the Fourier-space array are needed only implicitly and only during the FFT and inverse FFT operations. With most of the zero padding in the m direction eliminated and redundant terms unstored, all of the information in the FFT matrix can be stored in a $p \times 2p$ block. All the zero padding in the l direction is still required.

Another problem arising from the side effects of FFT convolution is the need to zero out coefficients outside of the allowed range of l and m after the M2L translation is computed. This prevents the full FMA from being computed entirely in Fourier space. This situation does not occur in the coefficient-space computation because $Y_l^m(\theta, \phi)$ equals zero outside of the allowed values of l and m, a property which has the effect of zeroing out M2L output coefficients values with indices outside of this range. In practice, the coefficient-space method of computing the M2L translation requires summation only over the allowed range of l, m, l', and m'. However, a convolution of the multipole expansion coefficients computed in Fourier space does not include any consultation with a spherical harmonic function, and therefore does not restrict values in that way. Rather, the convolution "smears" the coefficient matrices, resulting in an output coefficient matrix that contains many nonzero values outside the allowable range of l and m. Figure 3 diagrams the smearing effect of convolving an input coefficient matrix with a transfer function matrix both containing values for $l = 0$ to 3 and $m = -l$ to l with l running in the vertical direction and m running in the horizontal direction. Any further convolution on the output matrix, to perform an L2L translation, for example (the next step of the FMA algorithm following M2L), will give incorrect values unless the nonzero coefficients at the out-of-bounds values of l and m are eliminated.

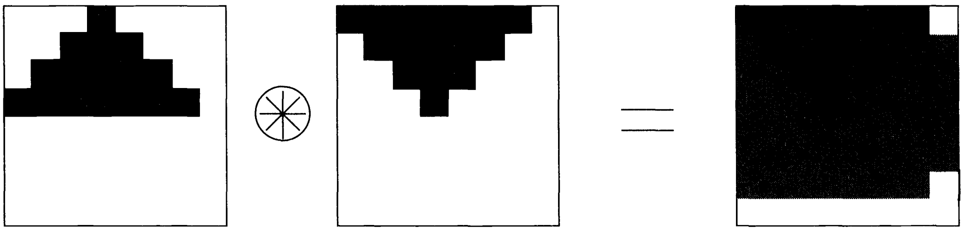


FIG. 3. *Convolution produces nonzero coefficients outside the allowed range of l and m. The figure shows input and output matrices for the* M2L *computation for $p = 4$ with the l values running in the vertical direction and the m values running in the horizontal direction. The dark regions show the nonzero values of the multipole expansion coefficient matrix and the transfer function matrix. In the output matrix, the dark regions show the locations of the desired data, within the allowed range of l and m, and the shaded regions show locations of nonzero values resulting from convolution that must be cleared.*

To zero out the coefficients where $l < 0$ or $|m| \geq l$, an element-by-element multiplication of the M2L coefficient-space output matrix with a mask matrix is required as follows:

$$\text{mask}(l, m) = \begin{cases} 1 & l \geq 0 \text{ and } -l \leq m \leq l, \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

Although this can be done in either coefficient space or Fourier space, it is much less expensive to do in coefficient space. This is because the operation is a direct multiplication of matrix elements in coefficient space but it is a convolution between the FFT of the above mask and the FFT of the matrix to be corrected. Incurring the overhead of converting out of Fourier space, multiplying and converting back to Fourier space is less expensive than convolving in Fourier space. This situation mirrors the reason for converting the M2L operation into Fourier space to start with, namely to avoid explicit convolution. The net result of the smearing effect of FFT convolution is that an inverse FFT (IFFT) must be performed on the M2L results prior to going on to perform L2L in Fourier space. In general, if all three operations, M2M, M2L, and L2L, are to be performed in Fourier space, an IFFT/multiply/FFT sequence must be performed between translations.

Although the coefficients must come out of Fourier space between translations to zero out the out-of-bounds coefficients, the results of each of the large number of M2L translations can be accumulated in Fourier space without any intermediate IFFT operations. This is important and valuable since the M2L is the most time-consuming portion of the FMA, and the repetition and accumulation reduces the overhead of converting the input multipole expansion coefficient matrix in and out of Fourier space to a small fraction of the M2L runtime. The transfer function matrix still requires an FFT for each M2L computation.

4.2. Numerical instability. Another difficulty in implementing the FFT version of the FMA is the numerical instability problem observed by Greengard and Rokhlin [10]. This instability comes from two sources. The first source is the nature of the warping operation on the multipole expansion and local expansion coefficients. These coefficients are warped with the A_l^m function, which ranges in magnitude from 1.0 to roughly $1/\sqrt{(2p)!}$. Thus, even for modest values of p, the range of magnitudes of values in the warped coefficient matrix exceeds the machine precision using double precision floating point numbers. The other source of instability arises when the dimensions of the problem are not scaled properly. The range of magnitudes due to r^n terms in the multipole expansion coefficients and the $1/\rho^n$ terms in the transfer function coefficients will cause numerical instability unless the dimensions in the problem are scaled to roughly within an order of magnitude of 1.0. Because the FFT operation produces a matrix of Fourier-space values that are linear combinations of the corresponding coefficient-space matrix values, combining input matrix values, which vary in magnitude, loses much of the information contained in the small magnitude numbers.

To make the algorithm numerically stable, Greengard and Rokhlin suggested a polynomial scaling scheme or a block decomposition of the coefficient matrices, and they implemented the polynomial scaling scheme. They reported comparable errors for the FFT and original versions of the FMA at $p = 8$ with single precision.

We implemented a block decomposition of the data for numerical stability and produced identical results to coefficient space computation out to $p = 32$ at double precision. We found that the block decomposition scheme provided many advantages including making the algorithms less sensitive to scaling and providing a speedup in performance by tailoring the size of the matrices to the nonzero values as described in the next section.

Block decomposition of the coefficient matrices in the l direction groups together coefficient values of proximate magnitude. This reduces the range of magnitudes input into the FFT function but requires several separate multiplications of matrix elements for each block. In other words, the block decomposition results in a large-grain convolution at the level of the coefficient blocks. In our implementation we selected a granularity of eight terms in the l direction (four coefficient terms and four zeros for padding) for each block to achieve a power of 2 for FFT efficiency and to reduce the numerical instability problems while keeping a large enough block size to minimize the block-level convolution. Figure 4 shows block-level decomposition for an example where $p = 16$. Each input block and transfer function block contains four rows of coefficients and four rows of zero padding. In the output blocks the smearing effect of convolution produces seven rows of useful data in the output blocks, except in the $l = 0$ to three block. We have demonstrated that this level of block decomposition of the coefficient matrices controls the numerical instability out to at least $p = 32$ for double precision data.

To demonstrate the numerical stability of the block decomposition scheme, we set up an experiment to compute the M2L translation between the two cells shown in Figure 5 with and without block decomposition at various scaling values, and we compared the results with coefficient-space computation. Each spherical, unit-radius cell contains 100 uniformly distributed particles. Each particle has a charge of -1.0 or 1.0, and the two values of charge occur with equal probability among the particles. The cell centers are separated by a distance of 4.0 along the vector $\vec{\rho}$, which leaves a cell diameter separation between the two cells. Because of the spherical symmetry, the results are essentially independent of the values of α and β. The results are shown in Figures 6 and 7 at single and double precision. The potential error for the N particles in cell B was computed by using (17) to compare the local expansion potential $\Phi^{\mathrm{B,local}}$ with the direct particle-to-particle potential Φ^{direct} computed at double precision. The theoretical worst-case error shown in the plots for this geometry is $\left(\frac{1}{2}\right)^p$. We have

$$\text{(17)} \qquad \text{Error} = \left(\frac{\sum_{i=1}^{N} (\Phi_i^{\mathrm{direct}} - \Phi_i^{\mathrm{B,local}})^2}{\sum_{i=1}^{N} (\Phi_i^{\mathrm{direct}})^2} \right)^{\frac{1}{2}} .$$

The s values in the plots indicate the values at which the dimensions of the cell and particle positions in Figure 5 were scaled. The scaling values were selected as powers of 2 in order to reflect the relative dimensions of cells encountered in the spatial decomposition of the FMA simulations region; at each level of decomposition, cells are cut in half in the x, y, and z directions to create eight child cells in the next level.

The single precision results with no block decomposition in Figure 6 show the polynomial scaling effect described by Greengard and Rokhlin [6, 9]. A scaling factor of 2–4 provides stability out to $p = 16$ by offsetting to some extent the $1/\sqrt{(2p)!}$ warping function with s^p

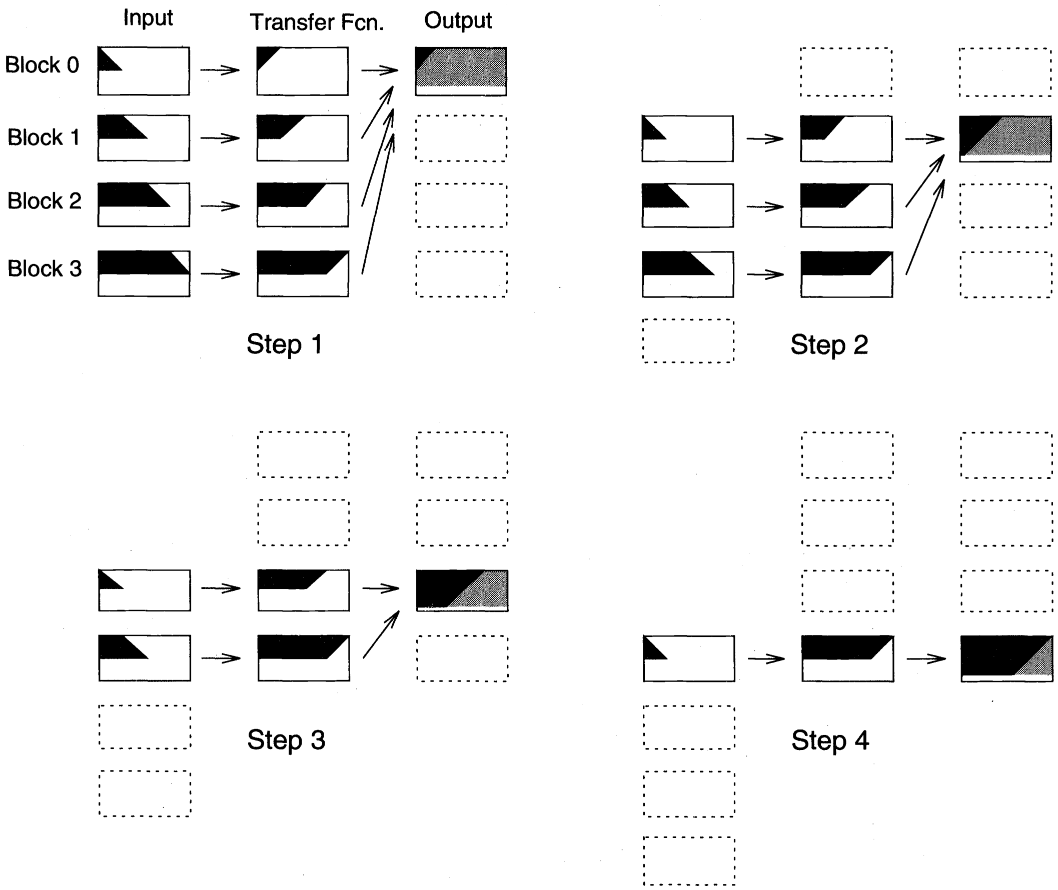


FIG. 4. *Large-grain convolution at the FFT block level. In this example* $p = 16$, *so decomposition of the coefficients into four blocks is required for numerical stability. Each input and transfer function block contains four rows of nonzero data corresponding to four* l *values. Only half of the coefficients are required in the* m *direction. The output blocks contain seven rows of useful data (except for block* 0) *plus some nonzero values shown in the shaded regions which are outside the allowed range of* l *and* m.

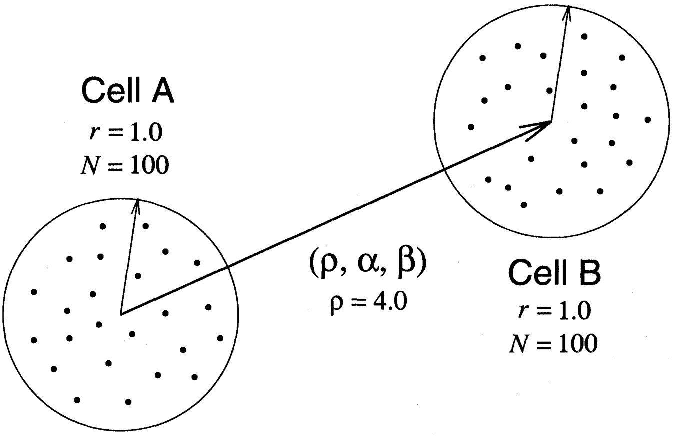


FIG. 5. *Geometry of the experiment to demonstrate numerical stability of FFT algorithms. Two spherical cells, A and B, each with radius* 1.0, *are separated by a cell diameter for a total separation of* 4.0 *measured at the cell centers. Each cell contains* 100 *charged particles, uniformly distributed, and any particle is equally likely to have a charge of* -1.0 *or* $+1.0$.

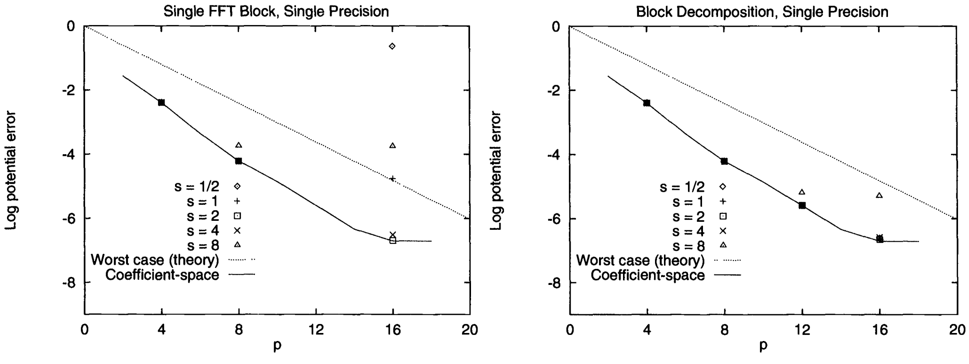


FIG. 6. *Results of the numerical stability experiment at single precision comparing the single block FFT computation of* M2L *(left) to the block decomposition scheme (right) with block size* 8. *For reference, both plots include the coefficient-space computation and the theoretical worst-case error* $\left(\frac{1}{2}\right)^p$. *The scaling factors s reflect the scaling of the hierarchical decomposition of the FMA, where each cell is halved in the x, y, and z directions to create* 8 *cells for the next level. The block decomposition scheme provides stability over a wide range of geometry scales.*

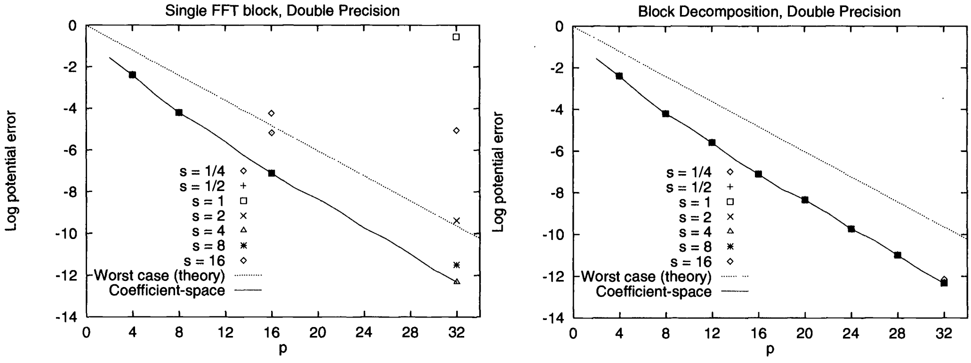


FIG. 7. *Results of the numerical stability experiment at double precision comparing the single block FFT computation of* M2L *(left) to the block decomposition scheme (right) with block size* 8. *Similar to the results in the previous figure, the block decomposition scheme provides stability over a wide range of geometry scales.*

polynomial scaling. The block decomposition method, on the other hand, is less sensitive to the scale of the geometry of the cells and particles and provides better stability over a wide range of scaling factors. Similarly, for the double precision case, the single block FFT provided stability when the geometries are carefully scaled, while the block decomposition FFT was stable over a wide range of scales. This block decomposition method thus allows all FMA translations to be computed at the scale determined by spatial decomposition, without rescaling each set of interactions.

4.3. More efficient block decomposition. In addition to improving the numerical stability, block decomposition provides a speedup opportunity due to the sparsity of the first few FFT array blocks. For example, in Figure 4, block 0 of the transfer function convolves only with input block 0. Both of these blocks could be as small as four units in width and when convolved together would still give the correct value of their contribution to output block 0. However, since block 0 of the input also convolves with the rest of the transfer function blocks, its width must be the full size p, which is 16 in the example of Figure 4. Block 0 of the transfer function, on the other hand, can be handled as if it is only four units wide, which is equivalent to making periodic replicas of the nonzero values of the transfer function block in the width of the full-size block.

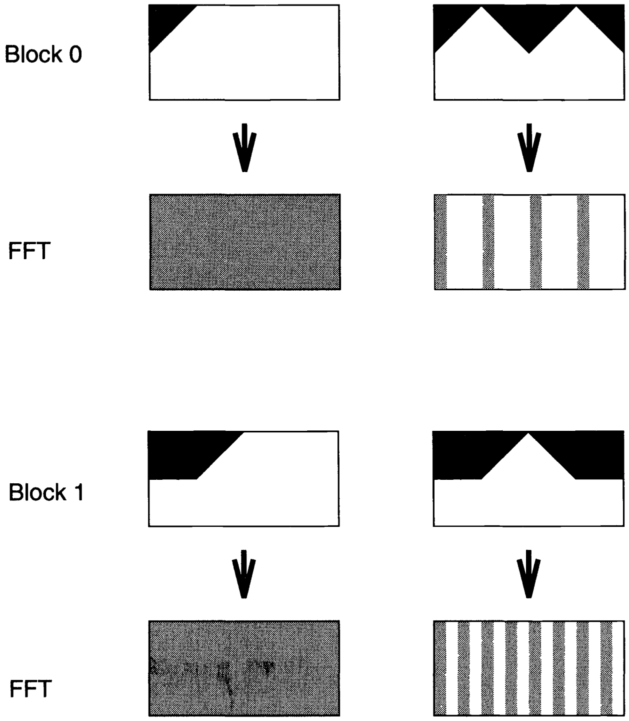


FIG. 8. *Simplifying the FFT of transfer function blocks. Blocks* 0 *and* 1 *from Figure* 5 *shown here handled as containing periodic replicas in the m direction to reduce the data required in the Fourier domain. Shaded areas of the Fourier transformed blocks show the locations of nonzero values.*

The advantage to modifying the effective transfer function block widths is in the Fourier transform of the block. As Figure 8 shows, the FFT of a function that is periodic in the m direction with a period that is a multiple of the FFT array length simplifies the FFT computation and produces fewer nonzero values for the subsequent matrix multiplication with the input blocks. Block 1 from Figure 4 can be treated in a similar fashion.

This modification to the transfer function block sizes improves the runtime of the FMA, because for each call to the M2L subroutine the transfer function is computed, transformed to the Fourier domain, and multiplied with a multipole expansion coefficient matrix. Shortening the effective width of the first few transfer function blocks lowers the computation count of the Fourier transforms and matrix multiplications involving those blocks.

The improved efficiency of the block decomposition method is clear from studying the operation count. Consider again the block decomposition in Figure 4 with $p = 16$. As described in §4.1, the fundamental size of the FFT matrix is $2p \times 2p$, although, because of the redundancy of the data in the m direction, only $p \times 2p$ are required for the matrix multiplication. The full width of the matrices in the m direction are required implicitly during FFT/IFFT operations, but only half of the matrices need to be stored. Furthermore, some of the blocks can be shortened (in Figure 4 the blocks 0 and 1) resulting in storage requirements diagrammed in Figure 9.

To analyze the operation count of the two principal computations in the M2L interaction, we consider the blocks to be twice their stored m direction width for the FFT, but equal to the stored width for the matrix multiplication. Since the full array is required in the m direction for the FFT, the nonstored values must be recovered from the stored values. Each block in

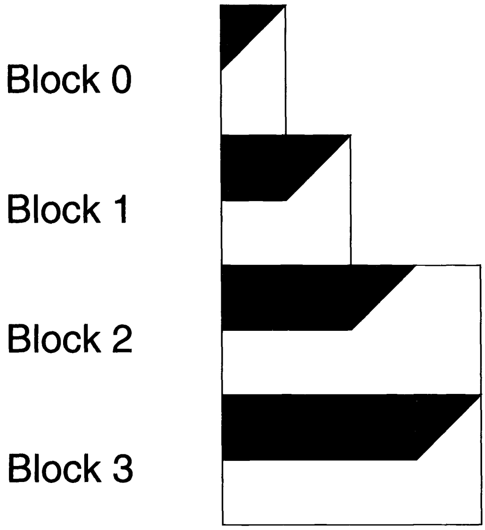


FIG. 9. *Storage requirements for shortened FFT blocks holding the transfer function coefficients. Dark regions show the location of the coefficients and light regions show zero padding.*

Figure 9 then requires $\frac{h(2w)}{2}(\log_2 h + \log_2 2w)$ complex multiply operations to compute the FFT, where h and w are the height and width of the individual block. The element-by-element matrix multiplication, on the other hand, requires only the information stored in the blocks; the remaining coefficients that make up the $2p \times 2p$ fundamental matrix are accounted for in the redundancy of the data or in the Fourier-space zeros of the periodic replica scheme described above. The number of complex multiplications for each block is simply hw.

As an example, dividing a coefficient matrix into four blocks and accumulating the computations required for the FFT of each block yields $p^2(\frac{11}{4}\log_2 p - \frac{1}{2})$ complex multiplications compared with $4p^2(\log_2 p + 1)$ computations for the FFT of the full $2p \times 2p$ matrix. The matrix multiplication requires slightly more computations for the block decomposition scheme because of the block-level convolution (Figure 4). For the case where the coefficients are decomposed into four blocks, $\frac{11}{4}p^2$ complex multiplications are required compared with $2p^2$ computations for the full matrix of $p \times 2p$ nonredundant values. For a concrete example, inserting $p = 16$ yields a total of 3,392 complex multiplications for the efficient block decomposition scheme which compares favorably to the 5,632 complex multiplications required for the FFT computation of M2L without block decomposition and 6,936 required for the original coefficient-space M2L computation. As is evident in §5 the operation count discussed above does not account for all of the speedup in the FFT method over the coefficient-space method. On one hand, for example, the regular locations of FFT block data allow more rapid computation on vector and pipelined machines. On the other hand, operations such as zeroing out the FFT matrix prior to each M2L uses nontrivial amounts of CPU time, and the extra memory usage required by the FFT version can result in more cache misses for large simulations run on machines with small caches. In general, however, the FFT version runs faster for any $p > 4$.

As an added benefit of the block decomposition method, p values other than powers of 2 can be selected with a smaller penalty in terms of extra zero padding. For example, $p = 12$ can be implemented with three blocks, each length 8 in the l direction. This eliminates the wasted zero padding in the l direction which would be required for a full 16×16 single block matrix to hold the $p = 12$ data.

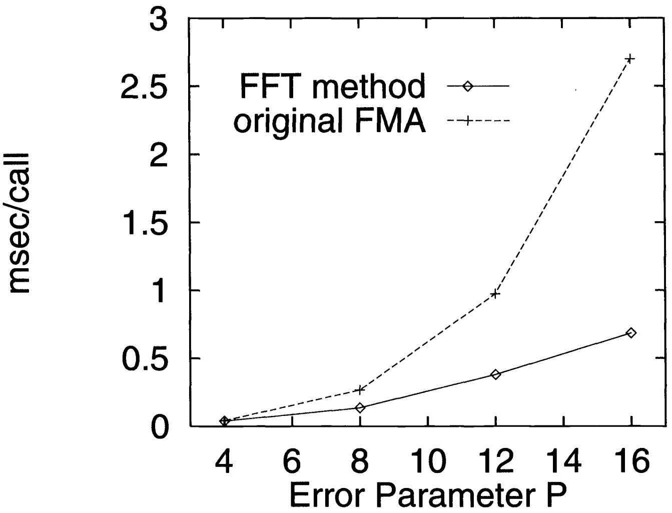


FIG. 10. M2L *subroutine performance comparison on IBM* RS-6000. *The msec/call times are an average reported by the UNIX prof utility running a simulation of* 20K *uniformly distributed charged particles.*

5. Implementation and results. Although it is possible to do the M2M and L2L translations in the Fourier domain as shown in the Appendix, our current implementation of the FFT version of the FMA does only the M2L translation in the Fourier domain. The M2M and L2L are done in coefficient space because their runtimes typically amount to only a few percent of the total. Also, the overhead of the FFT/IFFT conversion is amortized over fewer accumulated translations for M2M and L2L than for the M2L translation, so the speedup may not be as substantial.

Figure 10 shows the results of a comparison between the FFT version of the FMA with the M2L done in Fourier space and the original version of the FMA with the M2L done in coefficient space. The two versions have identical code from a locally generated FMA program except for the routines relating to the M2L translation. The plot shows the average time spent in the M2L subroutine as reported by the UNIX *prof* utility. The simulation is a 20K uniform distribution of charged particles, with an equal distribution of -1.0 and $+1.0$ charges. The simulation region is a cube in free space. The simulations were run on a single-processor IBM RS-6000 Model 360. The FFT routines used were derived from off-the-shelf code, optimized by fully unrolling the 1-D FFT portion of the routines and eliminating the unnecessary computations arising from the known zeros and data dependencies in the multipole expansion coefficients. On the RS-6000 the results of computing the M2L translation in the Fourier domain show a speedup factor of 2 measured in milliseconds per subroutine call (msec/call) at $p = 8$, and even higher speedup for p values greater than 8.

An additional benefit of conversion to Fourier space is the vectorizability of the code. Figure 11 shows the msec/call performance comparison for single-processor runs on a CRAY YMP-8 vector supercomputer, showing a nearly 3 times speedup at $p = 8$. Other computations in the M2L subroutine were not vectorized in this implementation (accounting for the faster msec/call time of the RS-6000 over the Cray), but this figure shows the improved vectorizability of the FFT method with no other changes to the code.

The runtime performance of any implementation of the FMA is dominated by two factors, only one of which is the subroutine calls to the M2L translations. While M2L dominates the

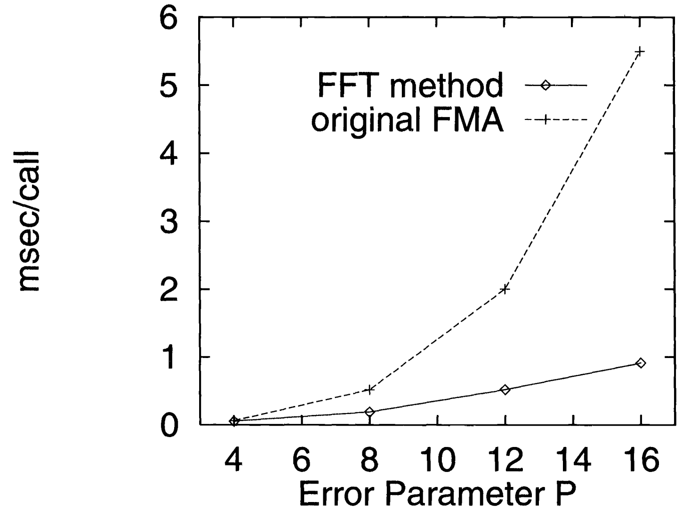


FIG. 11. M2L *subroutine performance comparison on Cray YMP-8. The simulation is the same as that used for the previous figure.*

multipole calculations, the direct, particle-to-particle interactions consume significant CPU time; these computations are necessary for regions where the multipole expansion series convergence criteria are not met. These regions are the cells at the leaf nodes of the oct-tree that are too close together to be put on the M2L interaction list. Particle-to-particle calculations in these regions are an independent task from the M2L cell-to-cell interactions, and the runtime of these calculations is dominated by the floating point square root calculation. In fact, extracting maximum performance from an FMA program for a given number of particles N and given p for desired accuracy typically results in roughly equal time spent between the particle-to-particle calculations and the M2L calculations. Thus, efforts to speed up the entire FMA algorithm by FFT conversion of the M2L are mitigated by the independent task of the particle-to-particle calculations. Figure 12 compares the speedup in peak particles per second (particles/sec) for various p values of the FFT version of the FMA code over the original version with the same particle-to-particle routines in both versions. The simulation is the same 20K system of uniformly distributed charged particles run on the single IBM RS-6000 processor described above.

6. Conclusion. In this paper we described a faster FMA, accelerated by converting the most common subroutine, the M2L translation, to the Fourier domain using an FFT. The result is an implementation that is $\mathcal{O}(p^2 \log_2(p)N)$. Conversion of the multipole expansion coefficients to the Fourier domain is satisfactory only after prewarping and zero-padding the coefficients. Block decomposition provides numerical stability to $p = 32$, and efficient block sizing reduces the overhead of the FFT enough to outperform coefficient-space convolution at low p values. Our implementation speeds up the FMA for $p > 4$, and speeds up the M2L subroutine call by a factor of 2 at $p = 8$ on an RISC processor and a factor of 3 at $p = 8$ on a vector processor.

To our knowledge, this is the first published version of the implementation details of an FFT version of the FMA, using the convolution form of the FMA computations described by Greengard and Rokhlin. The efficient block decomposition scheme described here is new. This block decomposition scheme provides numerical stability for the algorithm at double

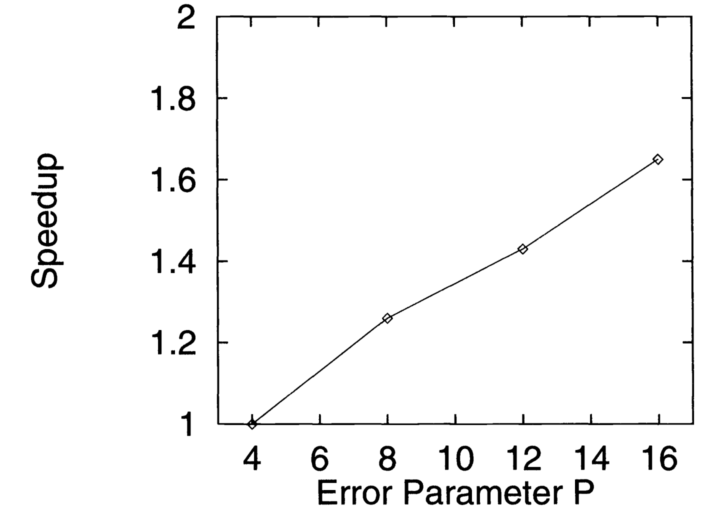


FIG. 12. *Speedup of a full FMA algorithm with FFT version of* M2L *on* RS-6000/360. *The simulation is the same as previous figures. The speedup of the multipole computations from the FFT enhancements is mitigated by the necessary particle-to-particle computations in the overall FMA algorithm.*

precision out to $p = 32$ over a wide range of geometry scales and provides an opportunity to restrict computations to the nonzero values in the multipole expansion coefficient matrix.

The speedup of the FMA described in this paper is particularly relevant to electrostatic applications. In gravitational computations, all the "charge" (mass) is positive, so the leading monopole term of the multipole expansion is large. Many astrophysical applications of multipole-based methods in fact retain only the monopole term in their expansions; i.e., they set $p = 1$. In electrostatics with approximate charge neutrality, the first moment of charge distributions will be nearly zero, so more terms must be retained in the expansions. Electrostatic applications typically felt the effect of the $\mathcal{O}(p^4)$ growth in runtime as terms were added to improve accuracy much more severely than gravitational applications. The runtime reductions achieved in this paper allow high accuracy electrostatic computations to be undertaken at moderate cost.

Appendix: FFT version of other FMA transformations. Multipole-to-multipole translation (M2M):

$$\Phi(\mathbf{r}) = \Phi'(\mathbf{r}') = \sum_{l,m}^{p-1} \mathbf{M}_l^m \frac{Y_l^m(\theta,\phi)}{r^{l+1}} = \sum_{l',m'}^{p-1} \mathbf{M}_{l'}^{m'} \frac{Y_{l'}^{m'}(\theta',\phi')}{r'^{l'+1}}, \tag{18}$$

$$\mathbf{M}_{l'}^{m'} = \sum_{l,m} \mathbf{T}^{M2M}_{l,m,l',m'} \mathbf{M}_l^m, \tag{19}$$

$$\mathbf{T}^{M2M}_{l,m,l',m'} = \frac{(-1)^{l'-l} A_l^m A_{l'-l}^{m'-m}}{A_{l'}^{m'}} Y_{l'-l}^{*m'-m}(\alpha,\beta)\rho^{l'-l}, \tag{20}$$

$$\left[(-1)^{-l'}\mathbf{M}_{l'}^{m'} A_{l'}^{m'}\right] = \sum_{l,m}\left[\mathbf{H}_{l'-l}^{m'-m}\right]\left[(-1)^{l}\mathbf{M}_l^m A_l^m\right], \tag{21}$$

$$[\mathbf{H}_l^m] = A_l^m Y_l^{*m}(\alpha, \beta)\rho^l, \tag{22}$$

$$y(l, m) = \left[(-1)^l \mathbf{M}_l^m A_l^m\right], \tag{23}$$

$$h(l, m) = [\mathbf{H}_l^m] = A_l^m Y_l^{*m}(\alpha, \beta)\rho^l, \tag{24}$$

$$x(l, m) = \left[(-1)^l \mathbf{M}_l^m A_l^m\right], \tag{25}$$

$$y(l', m') = \sum_{l,m} h(l' - l, m' - m)x(l, m) = h(l', m') \circledast x(l', m'), \tag{26}$$

$$y(l, m) = h(l, m) \circledast x(l, m) \leftrightarrow Y(\omega_l, \omega_m) = H(\omega_l, \omega_m)X(\omega_l, \omega_m). \tag{27}$$

Local-to-local translation (L2L):

$$\Phi(\mathbf{r}) = \Phi'(\mathbf{r}') = \sum_{l,m}^{p-1} \mathbf{L}_l^m Y_l^m(\theta, \phi) r^l = \sum_{l',m'}^{p-1} \mathbf{L}_{l'}^{m'} Y_{l'}^{m'}(\theta', \phi') r'^{l'}, \tag{28}$$

$$\mathbf{L}_{l'}^{m'} = \sum_{l,m} \mathbf{T}_{l,m,l',m'}^{L2L} \mathbf{L}_l^m, \tag{29}$$

$$\mathbf{T}_{l,m,l',m'}^{L2L} = \frac{A_{l'}^{m'} A_{l-l'}^{m-m'}}{A_l^m} Y_{l-l'}^{m-m'}(\alpha, \beta)\rho^{l-l'}, \tag{30}$$

$$\left[\frac{\mathbf{L}_{l'}^{m'}}{A_{l'}^{m'}}\right] = \sum_{l,m} \left[\mathbf{H}_{-(l'-l)}^{-(m'-m)}\right] \left[\frac{\mathbf{L}_l^m}{A_l^m}\right], \tag{31}$$

$$[\mathbf{H}_l^m] = A_l^m Y_l^m(\alpha, \beta)\rho^l, \tag{32}$$

$$y(l, m) = \left[\frac{\mathbf{L}_l^m}{A_l^m}\right], \tag{33}$$

$$h(l, m) = [\mathbf{H}_l^m] = A_l^m Y_l^m(\alpha, \beta)\rho^l, \tag{34}$$

$$x(l, m) = \left[\frac{\mathbf{L}_l^m}{A_l^m}\right], \tag{35}$$

$$y(l', m') = \sum_{l,m} h(-(l' - l), -(m' - m))x(l, m) = h(-l', -m') \circledast x(l', m'), \tag{36}$$

$$y(l, m) = h(-l, -m) \circledast x(l, m) \leftrightarrow Y(\omega_l, \omega_m) = H(\omega_{-l}, \omega_{-m})X(\omega_l, \omega_m). \tag{37}$$

Acknowledgments. The authors thank Loren Nolte for his expert instruction on the fast Fourier transform and discrete signal processing techniques.

REFERENCES

[1] J. Ambrosiano, L. Greengard, and V. Rokhlin, *The fast multipole method for gridless particle simulation*, Comput. Phys. Comm., 48 (1988), pp. 117-125.

[2] J. Barnes and P. Hut, *A hierarchical $O(N \log N)$ force-calculation algorithm*, Nature, 324 (1986), pp. 446-449.

[3] J. A. Board, Jr., R. R. Batchelor, and J. F. Leathrum, Jr., *High performance implementations of the fast multipole algorithm*, in Symposium on Parallel and Vector Computation in Heat Transfer, Proc. 1990 AIAA/ASME Thermophysics and Heat Transfer Conference, 1990.

[4] J. A. Board, Jr., J. W. Causey, J. F. Leathrum, Jr., A. Windemuth, and K. Schulten, *Accelerated molecular dynamics simulation with the parallel fast multipole algorithm*, Chemical Physics Letters, 198 (1992), pp. 89-94.

[5] J. Carrier, L. Greengard, and V. Rokhlin, *A fast adaptive multipole algorithm for particle simulations*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 669-686.

[6] L. Greengard, *The Rapid Evaluation of Potential Fields in Particle Systems*, MIT Press, Cambridge, MA, 1988.

[7] ———, *The numerical solution of the N-body problem*, Comput. Phys., 4 (1990), pp. 142-152.

[8] L. Greengard and W. Gropp, *A parallel version of the fast multipole method*, Comput. Math. Appl., 20 (1990), pp. 63-71.

[9] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, J. Comput. Phys., 73 (1987), pp. 325-348.

[10] ———, *On the efficient implementation of the Fast Multipole Algorithm*, Tech. Rep. RR-515, Dept. of Computer Science, Yale University, New Haven, CT, 1988.

[11] ———, *The rapid evaluation of potential fields in three dimensions*, in Vortex Methods, C. Anderson and C. Greengard, eds., Lecture Notes in Mathematics, Vol. 1360, Springer-Verlag, New York, Berlin, 1988, pp. 121-141.

[12] J. F. Leathrum, Jr. and J. A. Board, Jr., *The Parallel Fast Multipole Algorithm in Three Dimensions*, Tech. Rep., Dept. of Electrical Engineering, Duke University, Durham, NC, 1992.

[13] ———, *Unstructured Scientific Computation on Scalable Multiprocessors*, MIT Press, Cambridge, MA, 1992, pp. 161-178.

[14] V. Y. Pan, J. H. Reif, and S. R. Tate, *The power of combining the techniques of algebraic and numerical computing: Improved approximate multipoint polynomial evaluation and improved multipole algorithms*, in 32nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 92), IEEE, 1992, pp. 703-713.

[15] K. E. Schmidt and M. A. Lee, *Implementing the fast multipole method in three dimensions*, J. Statist. Phys., 63 (1991), pp. 1223-1235.

[16] F. Zhao and S. L. Johnsson, *The parallel multipole method on the connection machine*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 1420-1437.

SIAM J. SCI. COMPUT.
Vol. 17, No. 2, pp. 416–429, March 1996

COMPUTATION OF PSEUDO-DIFFERENTIAL OPERATORS*

GANG BAO† AND WILLIAM W. SYMES‡

Abstract. A simple algorithm is described for computing general pseudo-differential operator actions. Our approach is based on the asymptotic expansion of the symbol together with the fast Fourier transform (FFT). The idea is motivated by the characterization of the pseudo-differential operator algebra. We show that the algorithm is efficient through analyzing its complexity. Some numerical experiments are also presented.

Key words. pseudo-differential operators, fast Fourier transform, spatially varying filters, microlocal cut-off, data processing

AMS subject classifications. 35S05, 65T20, 86A22

1. Introduction. The theory of pseudo-differential operators (ΨDOs) has made many important contributions to the development of partial differential equations. It provides a natural way to decompose a differential operator, which may be difficult to study directly, into several pieces with a simple structure. A precise way to describe propagation of singularities for differential equations is in terms of ΨDOs. Pseudo-differential operators may be viewed as spatially varying filters with simple asymptotics at high frequencies. Pseudo-differential operators differentiate waves and wave-like signals according to directions of propagation. Pseudo-differential operators also arise naturally in diverse fields (often under different names!) such as wave propagation, electrical engineering, and geophysics.

Although the theory of ΨDOs, or, more generally, microlocal analysis, has been well established since the '60s, little attention appears to have been paid to the computation of ΨDOs. In this paper, we present a simple algorithm for the computation of general ΨDO actions. Our idea is based on the following characterization of ΨDOs.

Fact. The ΨDO algebra is generated by all differential operators and all powers of the Laplacian.

More precisely, ΨDOs and many functions of these (inverse, powers, ...) are included in ΨDOs in the high frequency asymptotic sense.

See Kohn and Nirenberg [3] for a detailed discussion.

2. Pseudo-differential operators. Here, we shall give a brief introduction to a class of ΨDOs. For a complete account of ΨDOs, as well as the calculus, the reader is referred to Taylor [5], Nirenberg [4], or Hörmander [2].

We begin with the introduction of the Fourier transform and the inverse Fourier transform. The Fourier transform acting on a "nice" function u defined in $\mathbf{R}^n$ is

$$\mathcal{F}(u) = \hat{u} = (2\pi)^{-n} \int u(x) e^{-ix\cdot\xi} dx,$$

and the inverse Fourier transform is defined by

$$\mathcal{F}^{-1}(u) = \int \hat{u}(\xi) e^{ix\cdot\xi} d\xi\ .$$

*Received by the editors November 5, 1993; accepted for publication (in revised form) October 17, 1994. This work was partially supported by the National Science Foundation grant DMS 89-05878, by the Office of Naval Research contract N00014-J-89-1115, by the Air Force Office of Scientific Research contract AFOSR 89-0363, and by the Geophysical Parallel Computation Project (State of Texas).

†Department of Mathematics, University of Florida, Gainesville, FL 32611-2082 (bao@math.ufl.edu).

‡Department of Computational and Applied Mathematics, Rice University, Houston, TX 77251-1892 (symes@rice.edu).

Remark. A number of algorithms for the numerical computation of (discrete) Fourier transforms have been made available. We shall use a version of the FFT in our numerical work. A detailed description may be found in Conte and De Boor [1]. See also Van Loan [6] for the most recent development in the field.

Pseudo-differential operators are usually defined in terms of symbols, which are smooth functions of both space and frequency variables satisfying certain estimates. More precisely, $q(x, \xi)$ is a member of the symbol class $S^m_{1,0}(\mathbf{R}^n)$ iff $q(x, \xi)$ is a smooth function and for any compact subset K of $\mathbf{R}^n$, and real α, β, there exists a constant $C_{K,\alpha,\beta}$, such that

$$|D^\alpha_x D^\beta_\xi q(x, \xi)| \le C_{K,\alpha,\beta}(1 + |\xi|)^{m-|\beta|} \tag{2.1}$$

for all $x \in K$ and $\xi \in \mathbf{R}^n$.

In this paper, we shall confine ourselves to a subclass of $S^m_{1,0}$, the class S^m, which is the most natural class and sufficient for many applications. A function $q(x, \xi)$ is in S^m if $q(x, \xi) \in S^m_{1,0}$ and there are smooth $q_{m-j}(x, \xi)$, homogeneous of degree $m - j$ in ξ for $|\xi| \ge 1$; i.e.,

$$q_{m-j}(x, r\xi) = r^{m-j} q_{m-j}(x, \xi), \quad |\xi| \ge 1\ , \quad r \ge 1 \tag{2.2}$$

such that

$$q(x, \xi) \sim \sum_{j \ge 0} q_{m-j}(x, \xi)\ , \tag{2.3}$$

in the sense that

$$q(x, \xi) - \sum_{j=0}^{N} q_{m-j}(x, \xi) \in S^{m-N-1}_{1,0}\ , \tag{2.4}$$

where $q_m(x, \xi)$ is called the principal symbol or principal part of $q(x, \xi)$ that carries the most important information about q.

Then the operator Q defined by

$$Q(x, D_x)u = \int q(x, \xi)\hat{u}(\xi)e^{ix\cdot\xi}d\xi \tag{2.5}$$

is called a ΨDO of order m or $Q \in OPS^m(\mathbf{R}^n)$.

In particular, differential operators with smooth coefficients are ΨDOs. Indeed, for such a differential operator of order m, the corresponding symbol is a polynomial in ξ of degree m, and consequently is a symbol in S^m. The asymptotic expansion of a symbol (2.3) is unique up to smoothing operators.

3. Algorithm. In this section we describe the algorithm explicitly. Its complexity will be examined in the section that follows. For the sake of simplicity, we shall only describe the idea of computing two-dimensional ΨDOs. Some obvious modifications may be made to compute ΨDOs of arbitrary dimension. Throughout, we shall always assume that the action of Q on u is meaningful. The precise conditions may be found in any one of the references [2]-[5].

Given a ΨDO $Q(x, z, D_x, D_z) \in OPS^m$ whose symbol is $q(x, z, \xi, \eta)$ and a function $u(x, z)$, our goal is to compute the action Qu efficiently. Let us assume that the asymptotic expansion of the symbol is given by

$$q(x, z, \xi, \eta) \sim \sum_{j \ge 0} q_{m-j}(x, z, \xi, \eta)\ , \tag{3.1}$$

where $q_{m-j}(x,z,r\xi,r\eta) = r^{m-j}q_{m-j}(x,z,\xi,\eta)$ for $(|\xi|, |\eta| \geq 1)$. Again, q_m denotes the principal symbol of q.

Knowing the asymptotic expansion of q we compute the ΨDO action Qu through computing $Q_{m-j}u$ for $j \geq 0$. We describe the calculation for $j = 0$, as it is representative. That is, we will describe an algorithm for computing the action of the principal part of a ΨDO. Evidently this algorithm could be applied recursively to compute general ΨDO actions. However, the principal part gives the dominant effect on high frequency inputs, which is most important for our intended applications. Thus, for us, computation of the principal part above is sufficient.

Let

$$\xi = \omega\cos\theta\ , \quad \eta = \omega\sin\theta\ , \quad \omega = \sqrt{\xi^2+\eta^2}\ . \tag{3.2}$$

The homogeneity of q_m in ξ and η yields

$$q_m(x,z,\xi,\eta) = q_m(x,z,\omega\cos\theta,\omega\sin\theta) = \omega^m\tilde{q}_m(x,z,\theta)\ , \tag{3.3}$$

where $\tilde{q}_m(x,z,\theta)$ is defined to be $q_m(x,z,\cos\theta,\sin\theta)$.

Since $\tilde{q}_m$ is periodic in θ, it has a Fourier series expansion as follows:

$$\begin{aligned}\tilde{q}_m(x,z,\theta) = \sum_{l=-\infty}^{\infty} c_l(x,z)e^{il\theta} &\simeq \sum_{l=-K/2}^{K/2} c_l(x,z)e^{il\theta}\\ &= \sum_{l=-K/2}^{K/2} c_l(x,z)(\cos\theta + i\sin\theta)^l\ ,\end{aligned} \tag{3.4}$$

where K is an indicator of the number of terms in the expansion.

It follows from the definition of ΨDO (2.5) that

$$\begin{aligned}Q_m u &\simeq \int\!\!\int d\xi d\eta e^{i(x\xi+z\eta)} \sum_{l=-K/2}^{K/2} \omega^m c_l(x,z)(\cos\theta+i\sin\theta)^l\hat{u}(\xi,\eta)\\ &= \sum_{l=-K/2}^{K/2} c_l(x,z)\int\!\!\int d\xi d\eta\omega^m e^{i(x\xi+z\eta)}(\cos\theta+i\sin\theta)^l\hat{u}(\xi,\eta)\\ &= \sum_{l=-K/2}^{K/2} c_l(x,z)\mathcal{F}^{-1}[\omega^{m-l}(\xi+i\eta)^l\hat{u}(\xi,\eta)],\end{aligned} \tag{3.5}$$

where, to obtain the last equality, we have used the relations $\cos\theta = \xi/\omega$ and $\sin\theta = \eta/\omega$ in (3.2). Observe that ω^{m-l} is the symbol of the $(m-l)/2$-power of the (negative) Laplacian, while ξ and η are symbols of differential operators $D_x = -i\partial_x$ and $D_z = -i\partial_z$, respectively.

The procedure implicit in the above formulae leads to an algorithm to evaluate $Q_m u$ approximately, as follows. Assume that u is sampled on a discrete grid,

$$\begin{aligned}U_{i,j} &= u(x_0+(i-1)\Delta x, z_0+(j-1)\Delta z)\ ,\\ &i = 1,\ldots,M,\ \ j = 1,\ldots,N\ ,\end{aligned} \tag{3.6}$$

with spacings Δx, $\Delta z > 0$. Assume similarly that a sampling of $\tilde{q}_m$ is given

$$\begin{aligned}Q_{i,j,k} &= \tilde{q}_m(x_0+(i-1)\Delta x, z_0+(j-1)\Delta z, k\Delta\theta)\ ,\\ i &= 1,\ldots,M,\ \ j = 1,\ldots,N,\ \ k = -K/2,\ldots,K/2\ .\end{aligned} \tag{3.7}$$

With $X = (M-1)\Delta x$, $Z = (N-1)\Delta z$, the sample rates in the frequency domain are $\Delta\xi = 1/X$, $\Delta\eta = 1/Z$, so the (unaliased) samples of the symbols of the Laplacian, D_x, and D_z are

$$\Omega_{p,r} = 2\pi\sqrt{(p\Delta\xi)^2 + (r\Delta\eta)^2}\ , \tag{3.8}$$

$$\Xi_{p,r} = 2\pi p\Delta\xi\ , \tag{3.9}$$

$$Z_{p,r} = 2\pi r\Delta\eta\ , \tag{3.10}$$

$$p = -M/2, \dots, M/2,\ r = -N/2, \dots, N/2\ ,$$

respectively.

PROCEDURE FOR COMPUTING $Q_m u$

```
1. Compute the discrete Fourier transform $\hat{U}$ of $U$.
2. For each $i \in \{1, \dots, M\}$ and $j \in \{1, \dots, N\}$, compute the discrete Fourier transform
   $\hat{Q}_{i,j} = \{Q_{i,j,l}\}_{l=-K/2}^{K/2}$ of $Q_{i,j} = \{Q_{i,j,k}\}_{k=-K/2}^{K/2}$.
3. Initialize $(QU)_{i,j} = 0.0$, for $i = 1, \dots, M$, $j = 1, \dots, N$.
   DO $l = -K/2,\ K/2$
     a  compute the inverse Fourier transform $\{R^l_{i,j}\}_{i=1,j=1}^{M,N}$ of
            $\Omega^{m-l}_{p,r}(\Xi_{p,r} + iZ_{p,r})^l \hat{U}_{p,r}$
        for $p = -M/2, \dots, M/2$ and $r = -N/2, \dots, N/2$.
     b  accumulate
            $(QU)_{i,j} = (QU)_{i,j} + \hat{Q}_{i,j,l}R^l_{i,j}$
   END DO
```

4. Complexity analysis. We return to the general case. The complexity will be analyzed by the number of multiplications. We also make a few remarks about the accuracy of the algorithm.

The direct method of computing the ΨDO action is by straightforward discretization of the definition

$$\begin{aligned} Q_m u &= \int\int q_m(x,\xi)\hat{u}(\xi)e^{ix\cdot\xi}d\xi \\ &= \mathcal{F}^{-1}[q_m(x,\xi)\hat{u}(\xi)]\ . \end{aligned}$$

Let us assume that the input function is discretized on a regular d-dimensional grid, as is the symbol q_m. We denote by N the number of grid points in each direction, assuming these are roughly similar. Assuming also that the discrete Fourier transforms are computed using an FFT algorithm. We then have the following result.

LEMMA 4.1. *The direct algorithm has $O(N^{2d}\log N)$ complexity.*

This is an immediate consequence of the well-known fact that the FFT exhibits $O(N\log N)$ complexity, where N is the length of the input sequence.

We next discuss the complexity of the new algorithm. For simplicity, we once again consider the two-dimensional case. The approximate complexity orders of the steps in the algorithm proposed above are

1. $N^2\log N$;
2. $N^2K\log K$; and
3. a. $KN^2\log N$, b. KN^2.

Hence the total complexity is $O(KN^2(\log N + \log K))$ in two dimensions.

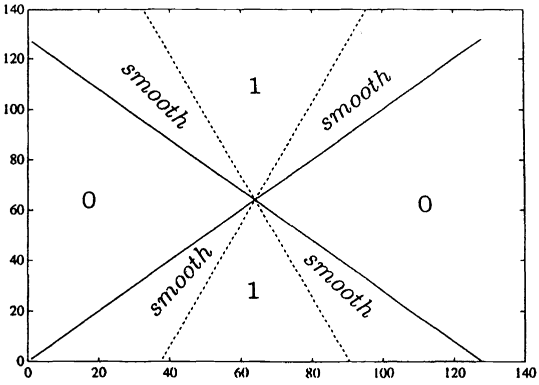


FIG. 1. *The symbol of a convolutional operator.*

In general, when the number of dimensions is d, a similar calculation will give Lemma 4.2.

LEMMA 4.2. *The new algorithm exhibits $O(K^{d-1}N^d(\log N + \log K))$ complexity.*

Remarks. The new algorithm is significantly superior to the direct method. In practice, the number of terms K in the finite θ-Fourier series approximation of q_m (3.4) ought to be chosen properly to make the sup-norm error small. If this is the case, the error in the computation of Q_m, modulo compact operators, will also be small (Taylor [5], p. 52). In particular the error will be small for oscillatory inputs u. Note that K is completely independent of N in this regard. Thus, in effect, the complexity of our algorithm is $O(N^d \log N)$!

In actual applications, N is usually big. When N and K are given, the numbers of multiplications involved in our algorithm and the direct method may be calculated easily. What makes the real difference is the number K. The theory and our numerical experiments both indicate that the number K depends only on the smoothness of the symbol, is insensitive to N, and can always be small.

5. Numerical experiments. In this section, we present the results of some numerical experiments carried out with the ΨDO algorithm. The class of ΨDO of great importance in our applications are microlocal cutoff operators, i.e., operators whose symbols are asymptotically one in some conic set and asymptotically zero in the complement of a slightly bigger conic set (essential support or aperture). These are the simplest undecomposable order zero ΨDOs. Our numerical experiments exhibit some interesting features of ΨDOs.

We begin with convolutional ΨDOs, which are ΨDOs that are independent of spatial variables. Convolutional operators are natural extensions of differential operators with constant coefficients. For this class of operators, it is easy to show that

$$\mathcal{F}(Qu) = Q(\xi)\hat{u}(\xi)\ . \tag{5.1}$$

This simple identity is useful in verifying the code. In fact, according to (5.1), one can recover the symbol from $\mathcal{F}(Qu)$ and $\hat{u}$. A symbol that characterizes a microlocal cutoff is specified by Figure 1, where the symbol is designed to be a C^2 function. Figure 2 displays the symbol function in terms of the angle θ. From this one-dimensional array, the ΨDO algorithm may be employed to compute the action, and hence the two-dimensional symbol function Q. The result, Figure 3, shows the symbol when the number of terms in Fourier series expansion of the symbol $K = 4$. It is easy to see that the symbol in Figure 3 illustrates the right direction

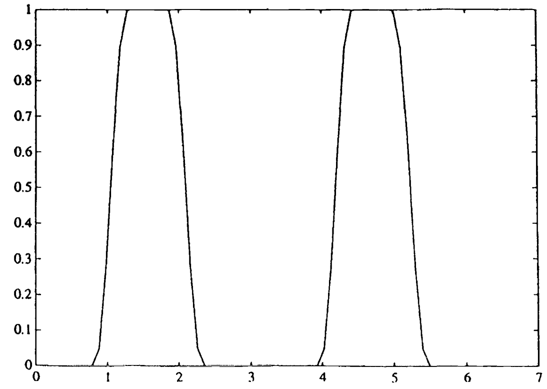
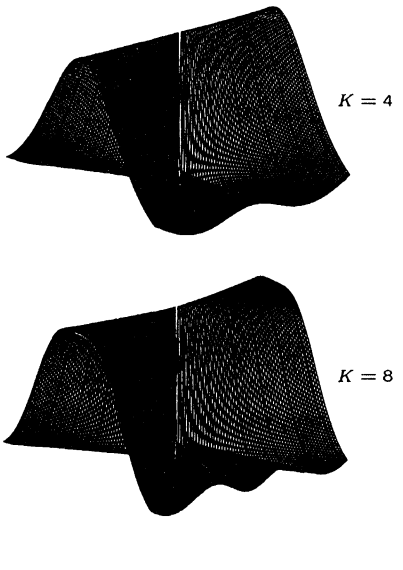


FIG. 2. *The same symbol as a function of θ: The angle with the horizontal axis.*



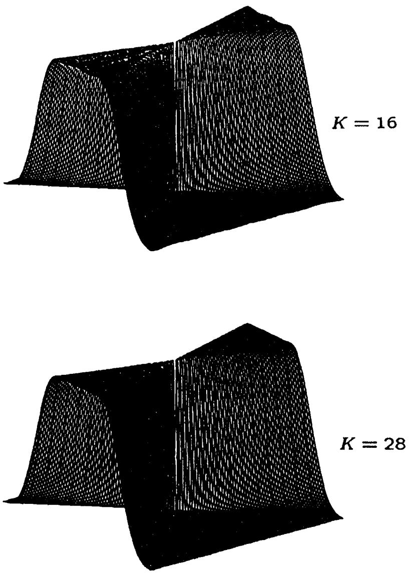


FIG. 3. *Symbol recovery*: $k = 4$.

FIG. 4. *Symbol recovery*: $k = 28$.

but wrong amplitude within the aperture. As the number of terms K increases, the recovery of the symbol becomes better and better. Figure 4 shows that the symbol is perfectly recovered after several steps. Again, we want to emphasize that the number K only depends on the smoothness of the symbol, and particularly is independent of the grid size N.

Another numerical experiment of ours concerns the rotation of apertures for convolutional operators. The function plotted in Figure 5 is a slightly smoothed characteristic function of a circle. We apply a ΨDO cutoff, whose symbol is given in Figure 6, to this function. Just as the theory predicts, the high frequency information of the resulting function (Figure 7) is well preserved within the aperture. We then rotate the symbol (Figures 8 and 10), and again the high frequency information is preserved in Figures 9 and 11, respectively. These examples are only illustrative, as the discrete Fourier transform allows a very simple and fast computation of convolution operators.

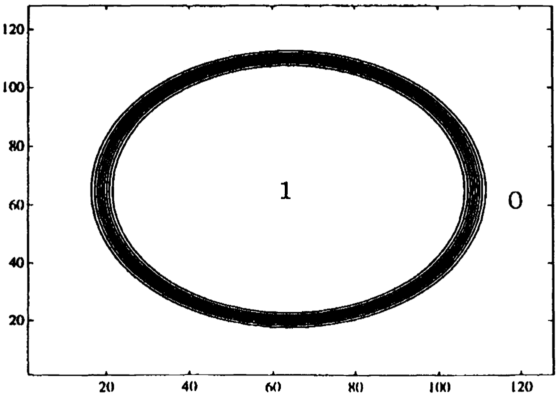


FIG. 5. *Function u.*

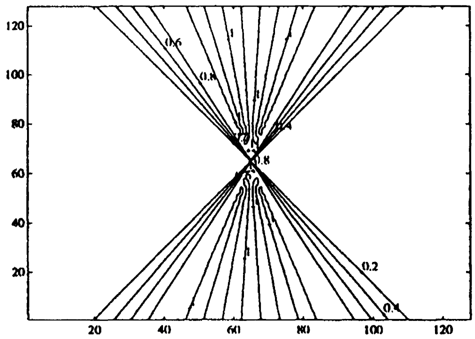


FIG. 6. *Symbol* q_0.

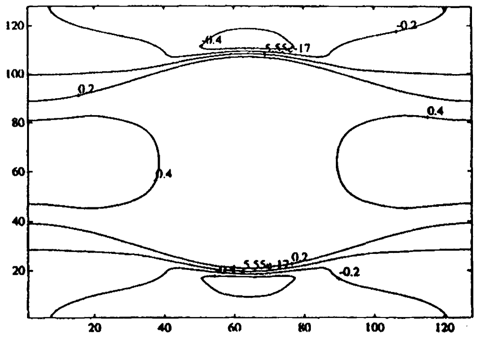


FIG. 7. *The action* $q_0 u$.

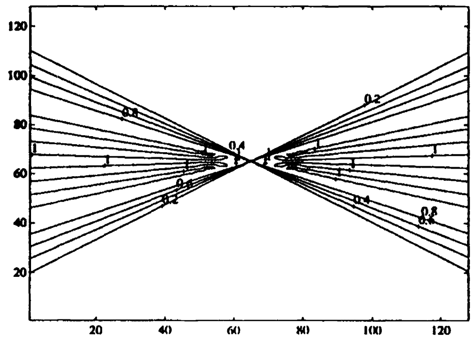


FIG. 8. *Symbol rotation* 1.

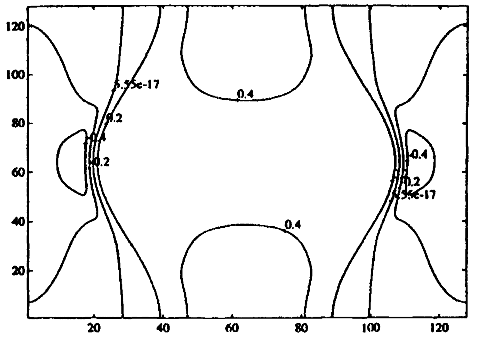


FIG. 9. *The action for rotation* 1.

Our next example is meant to illustrate the success of our algorithm with nonconvolutional ΨDOs. Figure 12 shows the symbol of a two-dimensional ΨDO, which is spatially varying (in the z-direction). The symbol can be generated from $q(z, \theta) = q_0(\theta + \delta\theta \sin(\pi z/z_{\max}))$, where q_0 is given in Figure 6, $\delta\theta$ is selected to be $\pi/2$, and $z \in [0, z_{\max}]$. Thus, as z increases, the symbol rotates smoothly; in particular the symbol will be equal to q_0 when z reaches its

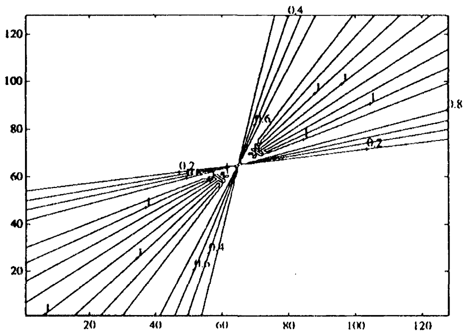


FIG. 10. *Symbol rotation* 2.

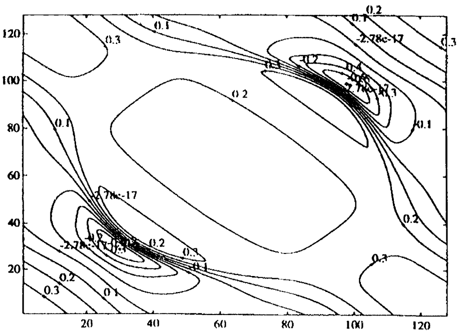


FIG. 11. *The action for rotation* 2.

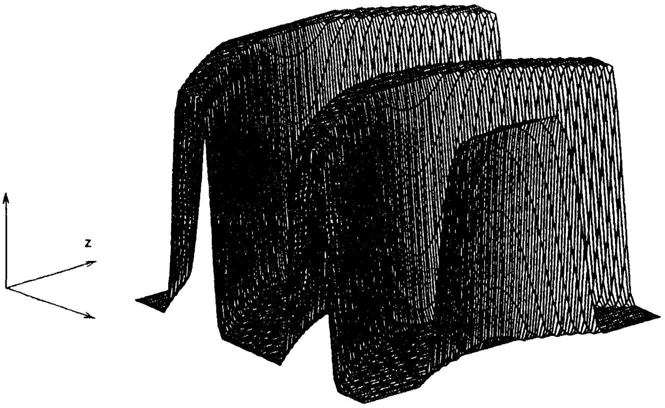


FIG. 12. *A spatially varying symbol.*

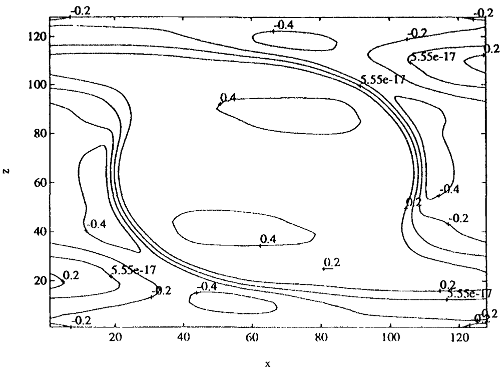


FIG. 13. *The action for the spatially varying symbol.*

maximum. Once again, the function u is the same as before in Figure 5. The result, as shown in Figure 13, agrees with the theory. Observe that the aperture is vertical for z near 0. The symbol rotates as z increases, so we start to see some high frequency horizontal components. When z is getting close to its maximum, the symbol rotates back, and the aperture becomes vertical again.

Our final example demonstrates an important application of the ΨDO algorithm to the seismic data processing in reflection seismology. The basic objective of all seismic processing is to convert the information recorded in a field into a form that most greatly facilitates geological interpretation of such a field. Evidently, real reflection data, which carry most of the information of the mechanical properties of the earth, are what geophysicists are most interested in obtaining through this process. Thus, an essential object of the processing is to eliminate or suppress all signals not associated with reflections. Figure 14 displays a seismogram, i.e., the recorded seismic data at receivers on the surface of the earth after an energy source is fired. The dark region that can be seen clearly contains very strong signals. These signals represent the early arrivals (direct and head waves). In the region below, there are other signals (reflections) which are not nearly as strong as the early arrivals. Unfortunately, the direct and head waves do not penetrate the earth; hence they contain no information about the subsurface about which we are interested. What contains useful information is the reflection energy in the lower region. This can be observed more clearly if one increases the amplitude of the seismogram as in Figure 15. The question arises: can one remove the early arrivals and yet keep the useful information of reflections? Applying the ΨDO computation algorithm, we design a microlocal cutoff (ΨDO) whose action on the seismogram is shown in Figure 16. The result appears to be very encouraging. The amplitude of the early arrivals is reduced dramatically, and meanwhile information of reflections is well preserved. We apply the same ΨDO filter to the data set once more to obtain an even better result shown in Figure 17. Now, the early arrivals are essentially gone, while again most of the reflections are preserved. We believe the noise left in the region where the early arrivals resided is caused by numerical scales; hence they can be eliminated. This processing technique is actually used in reflection seismology, where it is called "f-k dip filtering," e.g., Yilmaz [7], pp. 69–78. Our ΨDO algorithm yields an accurate and efficient means of "spatially variable dip filtering," for which we envision numerous uses. However, the enormous amount of computations of ΨDOs by the direct method makes any practical application impossible. Note that in the above examples, the grid size is taken to be 256 in each direction, which should be much bigger in real applications. However, the number K may be chosen much smaller than N due to the fact that K is independent of N.

6. Concluding remarks. A simple algorithm for the computation of a class of ΨDOs is introduced in this work. We exhibit some of the features of the algorithm. The complexity analysis indicates that the algorithm is much more efficient than the direct computation. Because of the simple structure, various massive parallel computers may be used to implement this algorithm so long as a fast FFT routine and fast array operations are available. In fact, some of our numerical experiments reported in this paper were obtained by using the Connection Machine.

We anticipate many applications of this algorithm. For example, ΨDOs are expected to play an important role in regularizing a class of ill-posed problems in multidimensional wave propagation that arise naturally in seismic inversion, oil and gas exploration, and many other related geophysical problems. Our experiment indicates the usefulness of microlocal (or ΨDO) cutoff in seismic data processing, i.e., sorting of waves according to direction in seismic data.

Mathematically, this algorithm should provide a way to compute the so-called microlocal norms of microlocal Sobolev spaces, which in turn would help us test the sharpness of various results on propagation of singularities for partial differential equations. This algorithm should also have some impact on signal processing, where ΨDOs form a class of spatially varying filters.

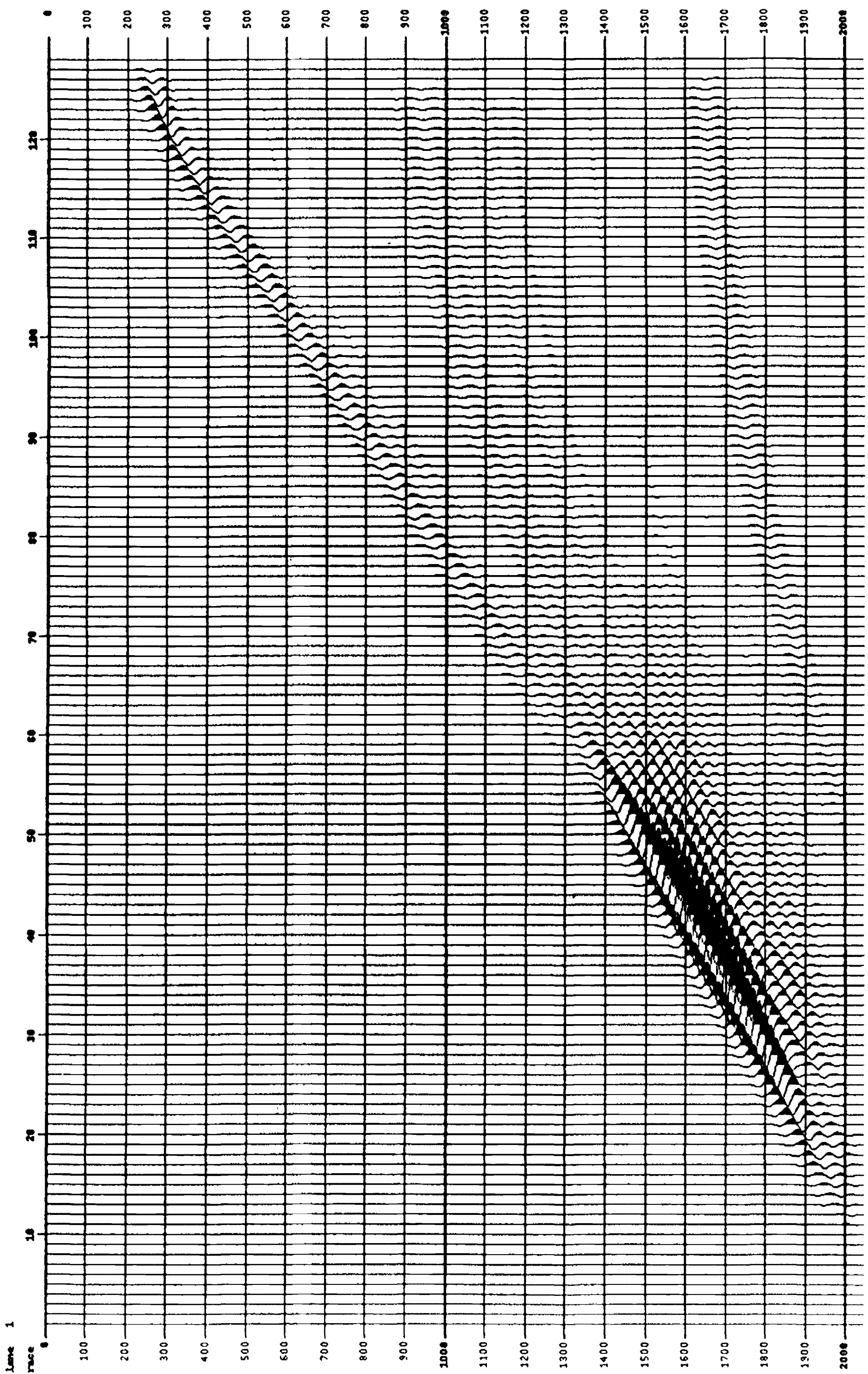


FIG. 14. *Seismogram.*

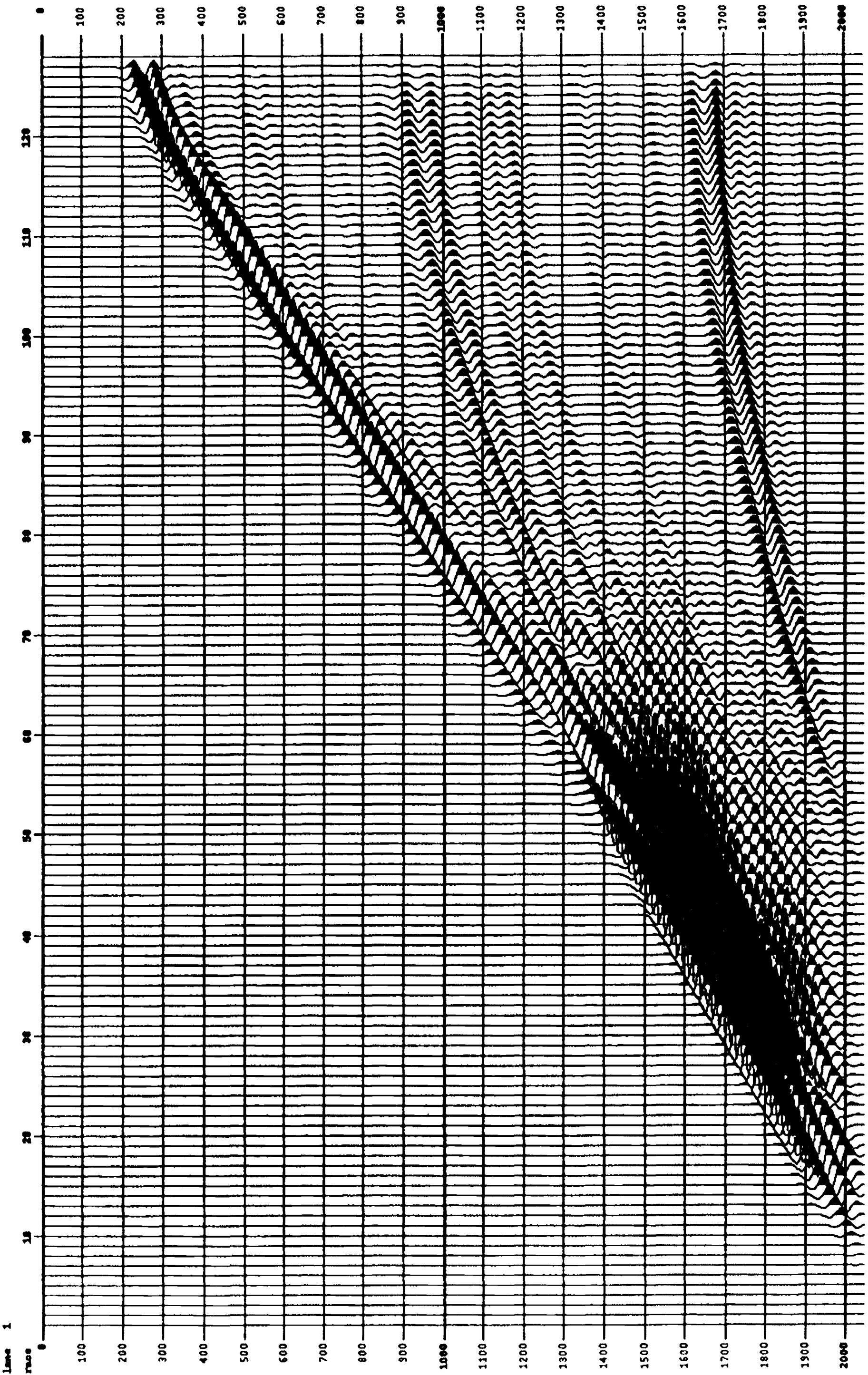


FIG. 15. *Seismogram with increased amplitude.*

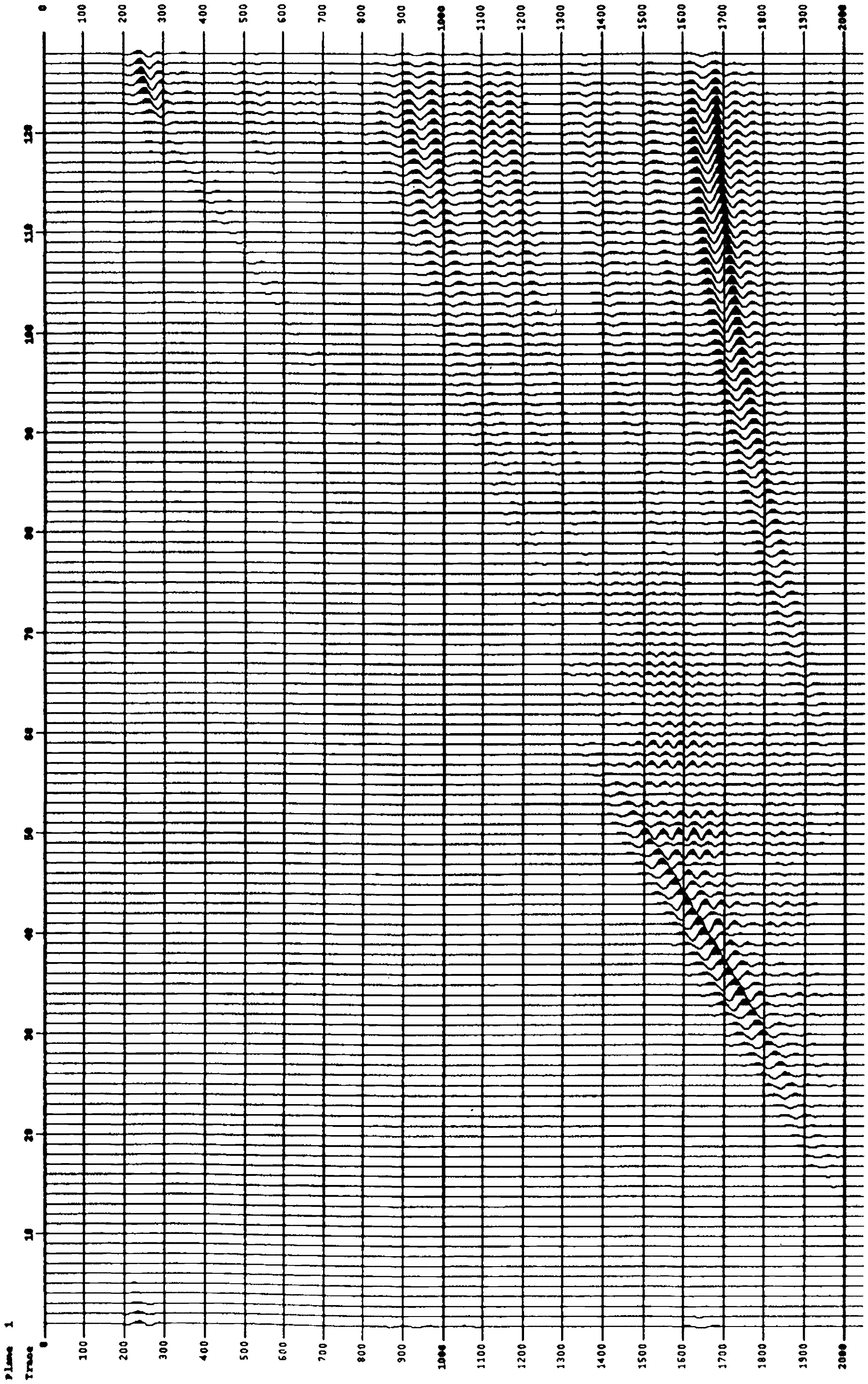


FIG. 16. *The result after applying the filter.*

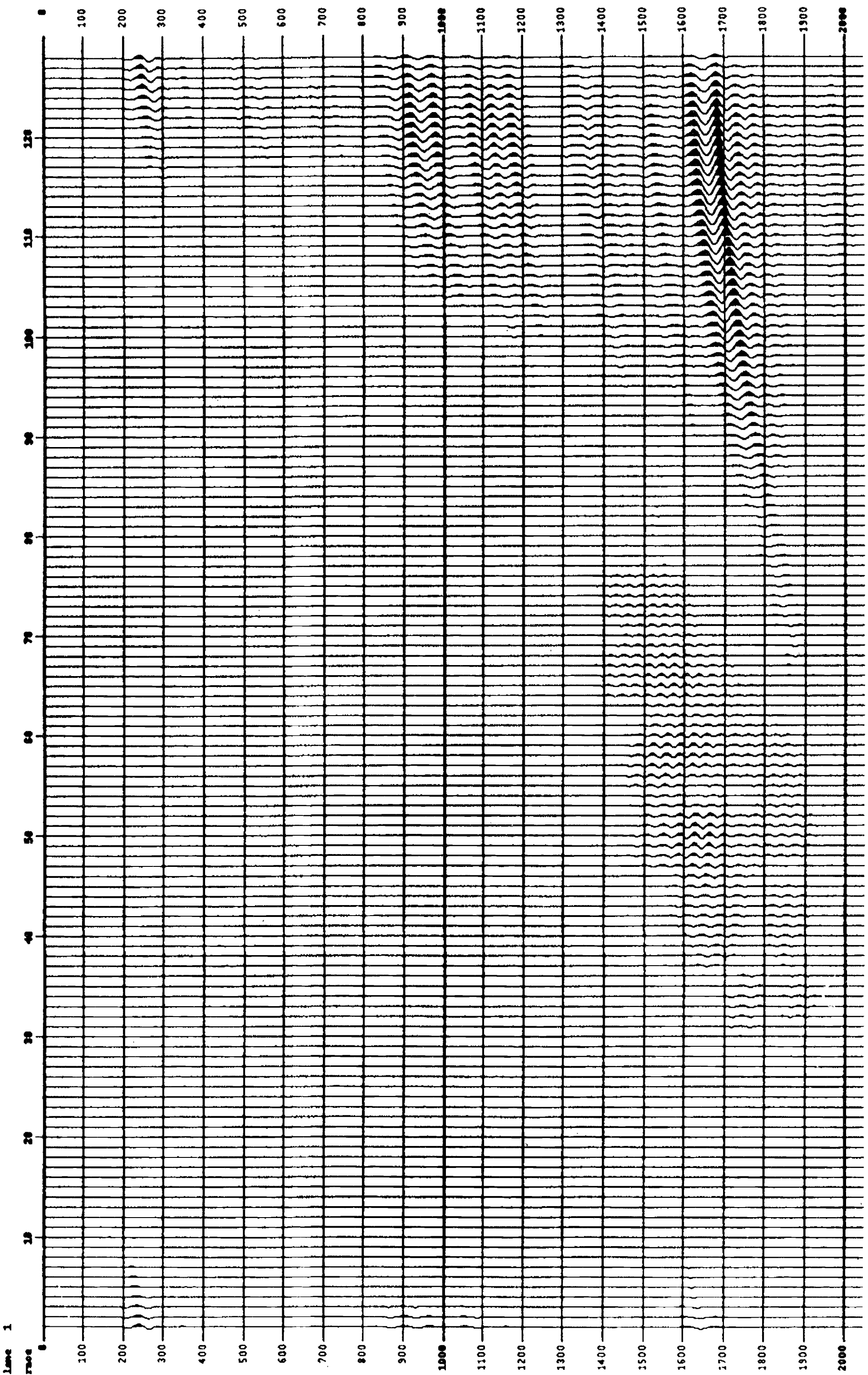


FIG. 17. *The result after applying the filter twice.*

REFERENCES

[1] S. D. Conte and C. De Boor, *Elementary Numerical Analysis*, 3rd ed., McGraw-Hill Inc., New York, 1980.

[2] L. Hörmander, *The Analysis of Linear Partial Differential Operators* III, Springer-Verlag, New York, 1985.

[3] J. J. Kohn and L. Nirenberg, *An algebra of pseudo-differential operators*, Comm. Pure Appl. Math., 18 (1965), pp. 269–305.

[4] L. Nirenberg, *Lectures on Linear Partial Differential Equations*, CBMS Regional Conf. Ser. in Math., No. 17, American Mathematical Society, Providence, RI, 1973.

[5] M. Taylor, *Pseudo-Differential Operators*, Princeton University Press, Princeton, NJ, 1981.

[6] C. Van Loan, *Computational Frameworks for the Fast Fourier Transform*, Society for Industrial and Applied Mathematics, Philadelphia, 1992.

[7] O. Yilmaz, *Seismic Data Processing*, Society of Exploration Geophysicists, Tulsa, OK, 1987.

SIAM J. SCI. COMPUT.
Vol. 17, No. 2, pp. 430-438, March 1996

BOUNDARY LAYER RESOLVING PSEUDOSPECTRAL METHODS FOR SINGULAR PERTURBATION PROBLEMS*

TAO TANG[†] AND MANFRED R. TRUMMER[‡]

Abstract. Pseudospectral methods are investigated for singularly perturbed boundary value problems for ordinary differential equations (ODEs) which possess boundary layers. It is well known that if the boundary layer is very small then a very large number of spectral collocation points is required to obtain accurate solutions. We introduce here a new effective procedure based on coordinate stretching and the Chebyshev pseudospectral method to resolve the boundary layers. Stable and accurate results are obtained for very thin boundary layers with a fairly small number of spectral collocation points.

Key words. spectral methods, singular perturbation, boundary layer

AMS subject classification. 65N35

1. Introduction.

We consider the pseudospectral (PS) method for the singular perturbation boundary value problem (BVP) given by

$$(1)\quad \epsilon u''(x) + p(x)u'(x) + q(x)u(x) = f(x), \quad x \in (-1,1), \quad u(-1) = \alpha, \;\; u(1) = \beta,$$

where $\epsilon > 0$ denotes a fixed (small) constant. In many applications, (1) possesses boundary layers, i.e., regions of rapid change in the solution near the endpoints, with widths $o(1)$ as $\epsilon \to 0$. It has been found that PS methods are attractive in solving this problem (see, e.g., [4]). By clustering the gridpoints toward the boundaries, for example, as in the Chebyshev method ($x_j = \cos\frac{\pi j}{N}$, $j = 0, 1, \ldots, N$), PS methods are more efficient than finite difference methods in resolving the boundary layers. However, in performance they still lag behind collocation methods with adaptive mesh selection (e.g., COLSYS [1]).

Although PS methods are remarkably accurate in exact arithmetic, there are a number of difficulties associated with its use. Especially with very small parameter ϵ in (1), large N is required to obtain accurate solutions (see, e.g., [11]). In addition, ill conditioning of the corresponding differentiation matrices with increasing N frequently causes degradation of the observed precision. Furthermore, as clarified in recent studies by Trefethen and Trefethen and Trummer [14, 15] the time step restrictions due to this ill conditioning can be more severe than those predicted by the standard stability theory, if such methods were to be applied to a time-dependent problem. Therefore, there has been considerable interest in recent years in developing well-conditioned spectral methods (see, e.g., [5-7]).

If $\epsilon \ll 1$ (e.g., $\epsilon < 10^{-6}$) and the problem possesses a boundary layer of width $O(\epsilon)$, high accuracy cannot be expected no matter how stable the spectral method is (see, e.g., [5, 11]). In the Chebyshev PS method, the spacing between the collocation points near the boundary is $O(N^{-2})$. For good resolution of the numerical solution at least one of the collocation points ought to lie in the boundary layer, which implies that $N = O(\epsilon^{-1/2})$. If $\epsilon = 10^{-8}$ then about 10^4 collocation points should be used, which is not practical in most calculations.

The Chebyshev spectral method and the finite difference method with coordinate stretching [8, 12] are two potentially useful methods for resolving the boundary layers. However,

*Received by the editors July 13, 1992; accepted for publication (in revised form) October 19, 1994.

[†]Department of Mathematics, Simon Fraser University, Burnaby, British Columbia V5A 1S6, Canada (tangtao@cs.sfu.ca). The research of this author was supported by Natural Science and Engineering Research Council of Canada grant OGP0105545.

[‡]Department of Mathematics, Simon Fraser University, Burnaby, British Columbia V5A 1S6, Canada (trummer@sfu.ca). The research of this author was supported by Natural Science and Engineering Research Council of Canada grant OGP0036901.

neither method works well if $\epsilon \ll 1$, since in this case N has to be very large. To avoid this difficulty we combine the two methods (with a new coordinate stretching technique) to solve (1). The idea is simple: first, the problem is replaced by an equivalent one using a transformation of the computational domain; second, the transformed problem is solved with the standard Chebyshev PS method. After the transformation more collocation points lie in the boundary layer than before, and there are collocation points in the layer even for fairly small values of N.

2. Transformations. As mentioned in §1, at least one of the collocation points should lie in a small neighborhood of $x = \pm 1$ in order to resolve the boundary layers. Therefore, we introduce a sequence of variable transformations so that there are some collocation points within distance ϵ from the boundaries ± 1 even for $\epsilon \ll 1$ and $N = O(10)$. These transformations are iterated SINE functions $x = g_m(y)$, $m = 0, 1, \ldots,$ where

$$(2) \qquad g_0(y) := y, \qquad g_m(y) = \sin\left(\frac{\pi}{2} g_{m-1}(y)\right), \quad m \geq 1.$$

The theorem below characterizes these transformations based on the relative spacing of the transformed Chebyshev-Gauss-Lobatto nodes.

THEOREM 2.1. *The following two statements hold for any integer $m \geq 0$.* (a) *The map g_m is one-to-one and $g_m([-1, 1]) = [-1, 1]$.* (b) *If $y_j = \cos(\frac{j\pi}{N})$, $j = 0, \ldots, N$, then*

$$g_m(y_0) - g_m(y_1) = g_m(y_{N-1}) - g_m(y_N) = \frac{8}{\pi^2}\left(\frac{\pi^2}{4N}\right)^{2^{m+1}} \left(1 + O(N^{-2})\right).$$

Proof. For (a) We need to show that $g'_m(y) \neq 0$ for $y \in (-1, 1)$, $|g_m(y)| \leq 1$ and $g_m(\pm 1) = \pm 1$, which can be proved by induction (see also (6)). To establish (b), we note that $g_0(y_0) - g_0(y_1) = \frac{\pi^2}{2N^2}\left(1 + O(N^{-2})\right)$. Assuming that

$$g_k(y_0) - g_k(y_1) = \frac{8}{\pi^2}\left(\frac{\pi^2}{4N}\right)^{2^{k+1}} \left(1 + O(N^{-2})\right)$$

and noting that $g_k(y_0) = g_{k+1}(y_0) = 1$, we obtain

$$\begin{aligned} g_{k+1}(y_0) - g_{k+1}(y_1) &= 1 - \sin\left(\frac{\pi}{2} g_k(y_1)\right) \\ &= 1 - \sin\left(\frac{\pi}{2}\left(1 - \frac{8}{\pi^2}\left(\frac{\pi^2}{4N}\right)^{2^{k+1}} \left(1 + O(N^{-2})\right)\right)\right) \\ &= 1 - \cos\left(\frac{4}{\pi}\left(\frac{\pi^2}{4N}\right)^{2^{k+1}} \left(1 + O(N^{-2})\right)\right) = \frac{8}{\pi^2}\left(\frac{\pi^2}{4N}\right)^{2^{k+2}} \left(1 + O(N^{-2})\right). \end{aligned}$$

Since $g_m(y_N) = -g_m(y_0)$ and $g_m(y_{N-1}) = -g_m(y_1)$, the proof of (b) is hereby complete. □

From Theorem 2.1 it can be expected that the transformations (2) together with the Chebyshev PS method can deal with extremely small boundary layers with a fairly small number of collocation points. For $m = 1, 2,$ and 3 (which correspond to one, two, and three SINE transformations), the distance between each boundary point and its nearest interior point is $O(N^{-4})$, $O(N^{-8})$, and $O(N^{-16})$, respectively. Therefore, even for very small ϵ such as $\epsilon = 10^{-12}$, at least one collocation point lies in the boundary layer even for moderate values of N, if two or three SINE transformations are used.

3. The transformed equations. We transform the singularly perturbed linear BVP (1) via the variable transformation $x \mapsto y(x)$ (or $x = x(y)$) into the new BVP

$$\epsilon v''(y) + P(y)v'(y) + Q(y)v(y) = F(y), \tag{3}$$

where v is the transplant of u, $v(y) = u(x(y))$. The transformed coefficients are

$$P(y) := \frac{p(x)}{y'(x)} + \epsilon \frac{y''(x)}{y'(x)^2}, \tag{4}$$

$$Q(y) := \frac{q(x)}{y'(x)^2}, \qquad F(y) := \frac{f(x)}{y'(x)^2}, \tag{5}$$

where again $x = x(y)$. It is clear from (3) - (5) that for any variable transformation $x \mapsto y(x)$ the two quantities $1/y'(x)$ and $y''(x)/[y'(x)]^2$ are of interest and should be easy to calculate.

We now consider the transformation $x = x(y) := g_m(y)$ of §2. In this case, the computation of $1/y'(x)$ is straightforward. Differentiating the recursion (2) we obtain

$$g_0'(y) = 1, \quad g_m'(y) = \frac{\pi}{2} \cos\left(\frac{\pi}{2} g_{m-1}(y)\right) g_{m-1}'(y), \quad m \geq 1. \tag{6}$$

Since $y'(x) = 1/g_m'(y)$, we have

$$\frac{1}{y'(x)} = \prod_{k=0}^{m-1} \left(\frac{\pi}{2} \cos\left(\frac{\pi}{2} g_k(y)\right)\right), \quad m \geq 1. \tag{7}$$

Now we define the functions $h_m(x)$, mapping $[-1, 1]$ onto itself, recursively via

$$h_0(x) := x, \qquad h_m(x) := \frac{2}{\pi} \arcsin\left(h_{m-1}(x)\right), \quad m \geq 1. \tag{8}$$

LEMMA 3.1. $h_m = g_m^{-1}$ *for* $m = 0, 1, \ldots$.

Proof. The case $m = 0$ is trivial. For $m \geq 1$, we let $z = h_m(g_m(y))$. It can be shown by induction that for $k = 0, \ldots, m$,

$$g_k(z) = h_{m-k}(g_m(y)). \tag{9}$$

For $k = m$ we therefore obtain

$$g_m(z) = h_0(g_m(y)) = g_m(y),$$

and, since g_m is injective, it follows $y = z$; i.e., $y = h_m(g_m(y))$. □

We now proceed to find a recursion for the quantity $h_m''(x)/[h_m'(x)]^2$. From (8) we obtain

$$\sin\left(\frac{\pi}{2} h_m(x)\right) = h_{m-1}(x), \qquad m \geq 1. \tag{10}$$

Differentiating (10) twice with respect to x yields

$$\frac{\pi}{2} \cos\left(\frac{\pi}{2} h_m(x)\right) h_m'(x) = h_{m-1}'(x), \tag{11}$$

$$-\left(\frac{\pi}{2}\right)^2 \sin\left(\frac{\pi}{2} h_m(x)\right) \left(h_m'(x)\right)^2 + \left(\frac{\pi}{2}\right) \cos\left(\frac{\pi}{2} h_m(x)\right) h_m''(x) = h_{m-1}''(x). \tag{12}$$

Finally, using (11) and (12) we obtain the recursion

$$\frac{h_m''(x)}{\left(h_m'(x)\right)^2} = \frac{\pi}{2} \tan\left(\frac{\pi}{2} h_m(x)\right) + \frac{\pi}{2} \cos\left(\frac{\pi}{2} h_m(x)\right) \frac{h_{m-1}''(x)}{\left(h_{m-1}'(x)\right)^2}. \tag{13}$$

Note that $h_0'(x) \equiv 1$ and $h_0''(x) \equiv 0$. Since $y(x) = h_m(x)$, the quantity $y''(x)/[y'(x)]^2$ can be computed easily using (13).

TABLE 1
Maximum errors for Example 1 ("-" *indicates the error is greater than* 10).

		$N = 32$	$N = 64$	$N = 128$	$N = 256$	$N = 512$
$\epsilon = 10^{-3}$	$m = 0$	4.39(00)	3.02(−01)	1.60(−04)	6.84(−14)	2.36(−13)
	$m = 1$	1.50(−01)	4.27(−04)	2.22(−11)	9.30(−14)	2.13(−13)
	$m = 2$	2.20(−02)	3.91(−04)	1.74(−09)	5.57(−12)	9.02(−11)
$\epsilon = 10^{-6}$	$m = 1$	-	8.37(00)	2.60(00)	1.14(−01)	2.32(−05)
	$m = 2$	4.77(00)	2.11(−01)	7.50(−03)	6.82(−07)	1.10(−10)
	$m = 3$	1.01(00)	1.73(−01)	2.50(−03)	2.59(−07)	1.08(−10)
$\epsilon = 10^{-9}$	$m = 1$	-	-	-	-	-
	$m = 2$	6.56(−01)	3.20(−01)	3.03(−01)	9.00(−02)	6.25(−5)
	$m = 3$	2.66(00)	9.11(−01)	2.33(−02)	3.06(−04)	1.08(−07)

4. Examples. We denote by Q_N the space of polynomials of degree $\leq N$. We collocate (3) at the Chebyshev-Gauss-Lobatto nodes $y_j = \cos\frac{j\pi}{N}$, $j = 1, \ldots, N-1$, leading to the PS method for (3) as follows: find $v_N \in Q_N$ such that

$$\epsilon v_N''(y_j) + P(y_j)v_N'(y_j) + Q(y_j)v_N(y_j) = F(y_j), \quad j = 1, \ldots, N-1; \tag{14}$$

$$v_N(-1) = \alpha, \quad v_N(1) = \beta. \tag{15}$$

To solve (14) and (15), we have to solve a matrix equation of the form $AV = b$, where $A \in \mathbf{R}^{(N-1)\times(N-1)}$ and $V, b \in \mathbf{R}^{N-1}$, with $V = (V_1, \ldots, V_{N-1})^T$. The $V_j = v_N(y_j)$ are approximations of $v(y_j)$. The matrix equation is solved in MATLAB, which uses the standard LINPACK routines.

Example 1. Our first example has variable coefficients and the solution develops two boundary layers of width $O(\epsilon)$ near $x = \pm 1$. The equation is

$$\epsilon u''(x) - xu'(x) - u(x) = f(x) = \left(\frac{x+1}{\epsilon} - 1\right)e^{\frac{x+1}{\epsilon}} - 2\left(\frac{x-1}{\epsilon} + 1\right)e^{\frac{x-1}{\epsilon}}, \tag{16}$$

where f is chosen such that the function

$$u(x) = e^{-\frac{(x+1)}{\epsilon}} + 2e^{\frac{x-1}{\epsilon}} \tag{17}$$

is an exact solution of the differential equation. The boundary conditions are $u(-1) = 1$ and $u(+1) = 2$. Note that function (17) will satisfy these boundary conditions to machine precision (machine epsilon equals $2.22 * 10^{-16}$ in double precision) for all values of $\epsilon \leq 0.05$.

This is a difficult problem since high resolution is needed to avoid oscillations in the middle of the interval. The Chebyshev PS method *without* transformation fails to resolve the solution satisfactorily for $\epsilon = 10^{-4}$, even with $N = 256$ (the maximum error, defined by $\max_j\{|v(y_j) - V_j|\}$, is approximately equal to 0.13 in this case, compared with errors of approximately 2×10^{-12} for $m = 1$ and $m = 2$). Table 1 contains the results of our experiments for $\epsilon = 10^{-3}$, $\epsilon = 10^{-6}$, and $\epsilon = 10^{-9}$.

Figure 1 shows the plot of the solution for $\epsilon = 10^{-9}$, $N = 256$, and $m = 3$, and Figure 2 shows the corresponding error. It may not come as a surprise to find the major portion of the error located in the middle of the interval since we have a coarser grid spacing there. However, it is interesting to note that in this case the strategy of moving points out of the region of large errors actually helps in the solution process. This indicates that a strategy for adaptive gridding will have to be rather sophisticated, as it would appear natural to move more points *into* the region exhibiting large errors.

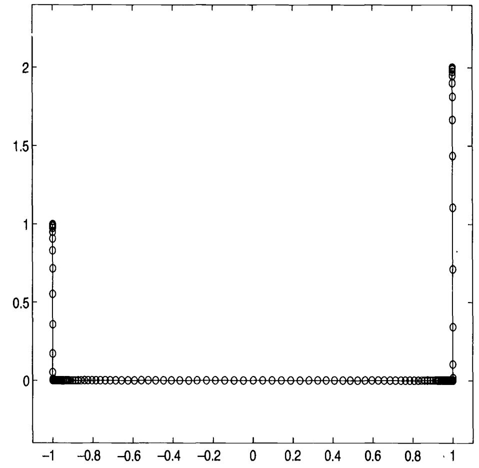


FIG. 1. *Numerical solution of Example* 1 *for* $\epsilon = 10^{-9}$, $N = 256$, *and* $m = 3$.

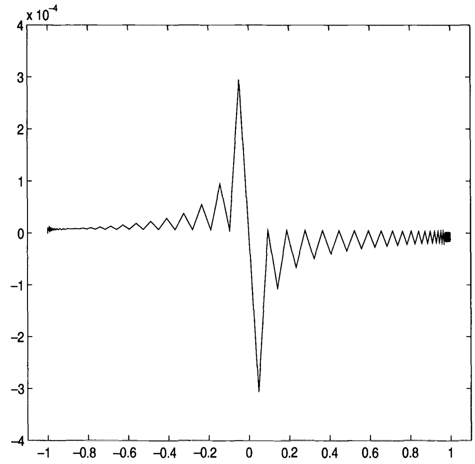


FIG. 2. *Error of Example* 1 *for* $\epsilon = 10^{-9}$, $N = 256$, *and* $m = 3$.

Example 2. Our second example is a nonlinear one; namely the stationary Burgers equation

$$\epsilon u''(x) + u(x)u'(x) = 0, \qquad x \in [-1, 1], \tag{18}$$

with boundary conditions chosen such that the function

$$u(x) = \tanh\left(\frac{x+1}{2\epsilon}\right) \tag{19}$$

is an exact solution. This function is 0 at the left boundary, and extremely close to 1 for most of the interval, with a boundary layer of width $O(\epsilon)$ at $x = -1$. The transformed equation with new variable $y = y(x)$ is simply

$$\epsilon v''(y) + \left[\frac{1}{y'(x)}v(y) + \epsilon\frac{y''(x)}{y'(x)^2}\right]v'(y) = 0, \qquad y \in [-1, 1]. \tag{20}$$

The solution is computed by Newton's method with $v \equiv 1$ as an initial guess; for small values of the parameter ϵ a continuation procedure for ϵ to obtain better initial guesses is advisable, and at times essential. Newton's method converges quickly and often in monotone (convergence problems during the first few iterations appear to indicate insufficient resolution of the discretization). Table 2 lists the results for $\epsilon = 10^{-3}$, $\epsilon = 10^{-6}$, and $\epsilon = 10^{-9}$.

A similar procedure has been applied to obtain numerical solutions for

$$\epsilon u''(x) + \frac{1}{2}u(x)u'(x) - \frac{1}{4}u(x) = 0, \qquad x \in [-1, 1], \tag{21}$$

with boundary conditions $u(-1) = u(1) = \frac{1}{2}$. This problem has the same type of nonlinearity as the stationary Burgers equation; it has been studied in detail in [9], and has been solved with COLSYS [2, pp. 382-383]. We find that for $\epsilon = 10^{-4}$ and $N = 64$ the method without transformation is developing oscillations near the boundary layer, whereas the approximation obtained with one SINE transformation ($m = 1$) easily resolves the boundary layer. Our results appear to be more accurate than the ones obtained with COLSYS for a comparable number of collocation points. In fact, with $m = 2$ we have no problem in resolving the boundary layer with $N = 128$ for ϵ as small as $\epsilon = 10^{-8}$ (see [13] for more details).

TABLE 2

Maximum errors for Example 2 (“$\star$” *indicates an error* > 1 *or convergence difficulties in the Newton process*).

		$N = 32$	$N = 64$	$N = 128$	$N = 256$
$\epsilon = 10^{-3}$	$m = 0$	$\star$	1.8144(−02)	3.6293(−04)	3.3776(−07)
	$m = 1$	3.5818(−02)	4.5561(−04)	1.3573(−07)	3.4528(−14)
	$m = 2$	1.6063(−02)	3.6709(−04)	6.3134(−08)	4.9238(−14)
$\epsilon = 10^{-6}$	$m = 1$	$\star$	$\star$	4.3554(−02)	1.3762(−03)
	$m = 2$	$\star$	2.5004(−02)	2.4636(−03)	9.7656(−07)
	$m = 3$	$\star$	3.7734(−02)	7.1848(−04)	2.3793(−07)
$\epsilon = 10^{-9}$	$m = 1$	$\star$	$\star$	$\star$	$\star$
	$m = 2$	$\star$	$\star$	$\star$	6.1103(−03)
	$m = 3$	$\star$	$\star$	6.7784(−03)	1.0602(−04)

5. Conditioning. Some recent work on spectral methods for BVPs is concerned with improving the condition numbers of the matrices for which linear systems have to be solved (e.g., [3, 5, 7]). Since the second-order Chebyshev differentiation matrix has a condition number $O(N^4)$, the corresponding linear systems quickly become very ill conditioned, even for moderate values of N. Interestingly enough, these large condition numbers do not appear to affect the accuracy in the solutions nearly as badly as one would expect. This was first observed by Berrut [3], who transformed the BVP to the circle and solved it with the much better conditioned Fourier spectral method, without seeing any improvement in the accuracy of solutions. However, the large condition numbers would be important in time stepping (so in this sense the PDE case is more difficult than the ODE case), or, if one were to solve the linear systems by iterative methods.

We would like to give a heuristic argument why our solutions are surprisingly accurate (we get close to machine precision, even in cases where the condition number of the linear system is approximately 10^8). Denoting the n-by-n matrix ($n = N - 1$) of our linear system by A, we compute the singular value decomposition $A = W\Sigma V^T$. Σ is a diagonal matrix with the singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$ on its diagonal. The singular vectors $v_1, v_2, \ldots,$ are the columns of V. Both V and. W are orthogonal n-by-n matrices. It is easy to see that the maximum magnification of roundoff errors in the right-hand side occurs when the exact solution u of the system is a multiple of the first singular vector v_1 and the perturbation δu is entirely in the direction of the last singular vector v_n. Figure 3 shows plots of four of the singular vectors[1] for the matrix A of Example 1, with $\epsilon = 10^{-2}$. Singular vectors belonging to large singular values are highly oscillatory, whereas singular vectors associated with small singular values are smooth (here, v_j has $n + 1 - j$ local extrema). This is not surprising, as A is a discretization of a differential operator, and therefore the statement above holds not only for Example 1. The exact solution has a substantial smooth component and roundoff errors cannot be expected to produce a completely smooth perturbation to the exact solution—on the contrary, a nonsmooth perturbation is much more likely to emerge. Thus, the actual amplification of the roundoff error is much smaller than the worst-case scenario of an amplification by $\text{cond}(A) = \sigma_1/\sigma_n$.

The condition numbers of the matrices generated by our repeated SINE transformations exhibit the same growth rates with N as the matrices for the original problem. The conditioning problem is largely unaffected by the transformation.

[1] The vectors are plotted against a stretched version of the interval (0, 1) to make the oscillations near the boundaries more visible.

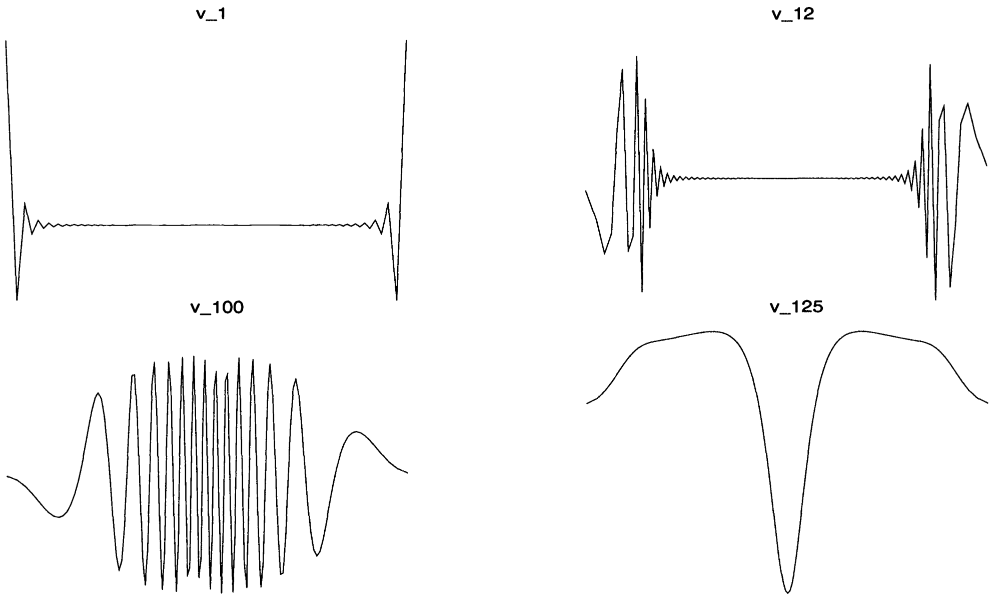


FIG. 3. *Singular vectors* v_1, v_{12}, v_{100}, *and* v_{125} *of* A *of Example* 1 *for* $\epsilon = 10^{-2}$, *plotted against a stretched interval* (0, 1).

Denoting again the Chebyshev-Gauss-Lobatto nodes by $y_j = \cos\frac{\pi j}{N}$, $j = 0, \ldots, N$, the first-order Chebyshev differentiation matrix D is given by

$$D_{kj} = \frac{c_k}{c_j}\frac{(-1)^{j+k}}{y_k - y_j}, \quad k \neq j, \tag{22}$$

$$D_{kk} = -\frac{y_k}{2(1-y_k^2)}, \quad k \neq 0, N, \tag{23}$$

$$D_{00} = -D_{NN} = \frac{2N^2+1}{6}, \tag{24}$$

where $c_k = 1$, except for $c_0 = c_N = 2$. It has been observed [4] that for large N the direct implementation of (22) - (24) suffers from cancellation, causing errors in the elements of the matrix D. Thus, it is advisable to replace (22) and (23) using trigonometric identities by the formulas

$$D_{kj} = \frac{c_k}{c_j}\frac{(-1)^{j+k}}{\sin((k+j)\pi/(2N))\sin((k-j)\pi/(2N))}, \quad k \neq j, \tag{25}$$

$$D_{kk} = -\frac{y_k}{2\sin^2(k\pi/N)}, \quad k \neq 0, N. \tag{26}$$

Finally, to avoid computing the SINE of arguments larger than $\pi/2$ in absolute value, one can take advantage of the symmetry property

$$D_{N-k,N-j} = D_{kj}. \tag{27}$$

Thus the most accurate method of computing D is using formulas (25) - (26) to find the *upper left triangle* of D (i.e., compute D_{kj} with $k + j \leq N$), and then use relation (27) for the other elements. This also appears to be more efficient (at least in our MATLAB implementation).

It should be noted that the effect of a more accurate matrix D cannot always be felt. To be noticeable, N has to be quite large, and the approximate solution must be extremely accurate. For Example 1 (see §4) with $\epsilon = 10^{-2}$, $N = 128$, and $m = 2$, the maximum error is $1.29 * 10^{-14}$, if the more accurate D is used, whereas the error with D computed by formulas (22)-(24) is $1.22 * 10^{-13}$.

6. Conclusions. Very thin boundary layers still must have one or more collocation points within the boundary layer. This results in extremely fine discretizations if the relative spacing of the gridpoints remains unchanged. Although the Chebyshev PS methods are more efficient than finite difference methods in resolving boundary layers, for $\epsilon \ll 1$ they still may need extremely large N to produce reasonable results, as discussed in §1. A much better approach for resolving the boundary layer is to use a mapping. However, a single mapping such as that of [8] is often not sufficient when $\epsilon \ll 1$.

To obtain good resolution for boundary layer problems, at least one of the grid points should lie in the boundary layer no matter how small the boundary layer is. The iterated SINE transformations introduced in §2 provide a very useful coordinate stretching technique to achieve this goal. Theoretically, as indicated in Theorem 2.1, these particular transformations together with the Chebyshev PS method can deal effectively with very small boundary layers using only a fairly small number of collocation points. Even for very small ϵ such as $\epsilon = 10^{-9}$, two or three SINE transformations with $N \approx 100$ are found to be sufficient to resolve the boundary layer, while most of the previously reported finite difference or spectral calculations cannot handle the case when ϵ is as small as 10^{-9}.

Section 3 of this paper gives a practical procedure for implementing the transformations. The transformation technique is also successful for nonlinear BVPs whose solutions have boundary layers. To date the most reliable methods for solving two-point BVPs are based on the collocation method with adaptive mesh selection (e.g., COLSYS [1, 2]). However, for boundary layer problems the present method is a serious competitor, in particular when spectral accuracy is a desirable feature.

The ill conditioning of the linear systems to be solved does not appear to be a serious problem as our experiments and the heuristic argument in §5 indicate. However, care must be taken if one uses these matrices in explicit time stepping in the time-dependent case, or, in the ODE case, if iterative methods are employed to solve the linear system.

Many practical problems possess boundary layers. For example, viscous flows have boundary layers next to solid surfaces where the tangential velocity is reduced to zero. The use of the finite difference method or the Chebyshev PS method is expensive for high Reynolds number flows. The numerical technique introduced in this work can be applied to solve more practical problems (see, e.g., [10]).

REFERENCES

[1] U. M. Ascher, J. Christiansen, and R. D. Russell, *A collocation solver for mixed order systems of boundary value problems*, Math. Comp., 33 (1979), pp. 659-679.

[2] U. M. Ascher, R. M. M. Mattheij, and R. D. Russell, *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*, Prentice-Hall, Englewood Cliff, NJ, 1988.

[3] J. P. Berrut, *A pseudospectral Čebyšev method with preliminary transform to the circle: Ordinary differential equations*, Technical rep. 252, Math. Institut, Techn. Universität München, Germany, 1990; private communication, 1989.

[4] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang, *Spectral Methods in Fluid Dynamics*, Series of Computational Physics, Springer-Verlag, Heidelberg, Berlin, New York, 1988.

[5] H. Eisen and W. Heinrichs, *A new method of stabilization for singular perturbation problems with spectral methods*, SIAM J. Numer. Anal., 29 (1992), pp. 107-122.

[6] B. Fornberg, *An improved pseudospectral method for initial-boundary value problems*, J. Comput. Phys., 91 (1990), pp. 381-397.

[7] L. Greengard, *Spectral integration and two-point boundary value problem*, SIAM J. Numer. Anal., 28 (1991), pp. 1071-1080.

[8] E. Kalnay de Rivas, *On the use of nonlinear grids in finite-difference equations*, J. Comput. Phys., 10 (1972), pp. 202-210.

[9] J. Kevorkian and J. Cole, *Perturbation Methods in Applied Mathematics*, Springer-Verlag, Berlin, 1981.

[10] W. Liu and J. Shen, *A new efficient spectral Galerkin method for singular perturbation problems*, preprint, 1994 (submitted).

[11] Y. Y. Liu, *The pseudospectral Chebyshev method for two-point boundary value problems*, master's thesis, Dept of Mathematics and Statistics, Simon Fraser University, Burnaby, B.C., Canada, 1992.

[12] S. A. Orszag and M. Israeli, *Numerical simulation of viscous incompressible flows*, Ann. Rev. Fluid Mech., 6 (1974), pp. 281-318.

[13] T. Tang and M. R. Trummer, *Boundary layer resolving pseudospectral method for singular perturbation problems*, Research report 93-06, Dept of Mathematics and Statistics, Simon Fraser University, B.C., Canada, 1993.

[14] L. N. Trefethen, *Lax-stability vs. eigenvalue stability of spectral methods*, in Numerical Methods for Fluid Dynamics III, K. W. Morton and M. J. Baines, eds., Clarendon Press, Oxford, 1988.

[15] L. N. Trefethen and M. R. Trummer, *An instability phenomenon in spectral methods*, SIAM J. Numer. Anal., 24 (1987), pp. 1008-1023.

SIAM J. SCI. COMPUT.
Vol. 17, No. 2, pp. 439-453, March 1996

TOWARDS AUTOMATIC MULTIGRID ALGORITHMS FOR SPD, NONSYMMETRIC AND INDEFINITE PROBLEMS*

YAIR SHAPIRA†, MOSHE ISRAELI†, AND AVRAM SIDI†

Abstract. A new multigrid algorithm is constructed for the solution of linear systems of equations which arise from the discretization of elliptic PDEs. It is defined in terms of the difference scheme on the fine grid only, and no rediscretization of the PDE is required. Numerical experiments show that this algorithm gives high convergence rates for several classes of problems: symmetric, nonsymmetric and problems with discontinuous coefficients, nonuniform grids, and nonrectangular domains. When supplemented with an acceleration method, good convergence is achieved also for pure convection problems and indefinite Helmholtz equations.

Key words. convection-diffusion equation, discontinuous coefficients, elliptic PDEs, indefinite Helmholtz equation, automatic multigrid method

AMS subject classifications. 65F10, 65N22, 65N55

1. Introduction.

The multigrid method is a powerful tool for the solution of linear systems which arise from the discretization of elliptic PDEs [4], [5]. In a multigrid iteration the equation is first relaxed on a fine grid in order to smooth the error; then the residual equations are transferred to a coarser grid, to be solved subsequently and to supply correction terms. Recursion is used to solve the coarser grid problem in a similar way. In order to implement this procedure the PDE has to be discretized on all grids and restriction and prolongation operators must be defined in order to transfer information between fine and coarse grids. The basic multigrid method works well for the Poisson equation in the square, but difficulties arise with nonsymmetric and indefinite problems and problems with variable coefficients, complicated domains, or nonuniform grids. In these cases, an effective discretization of the PDE on coarse grids becomes more complicated than that provided by a naive approach. Some suggestions on handling discontinuous coefficients are given in [1], while the nonsymmetric case is analyzed in [9] and [10]. A projection method for the solution of slightly indefinite problems is developed in [7]. Another projection method for such problems is presented and analyzed in [3]. These approaches, however, involve specialized and problem-dependent treatment, and the need for a uniform approach is not yet fulfilled.

Present multigrid procedures are not able to serve as "black box" solvers. Special attention has to be given to the neighborhood of the boundary and to the presence of discontinuities. In [6], [20], and [21] the algebraic multigrid (AMG) method is developed. This method is algebraic in the sense that it depends on the discrete system of equations and not on the original PDE or the difference scheme for it. It automatically chooses the coarse level variables and constructs the coarse level equations and the restriction and prolongation operators; hence it applies to general linear systems of equations. However, the set-up time required is large (equivalent to about 10 V-cycles).

Multigrid versions whose definition depends on the difference scheme on the original grid only also exist; these methods, which are called automatic methods in the sequel, reduce the original grid to further coarse grids and automatically construct the coarse-grid coefficient matrix and the restriction and prolongation operators. The black-box multigrid method of [11] applies to problems with discontinuous coefficients and nonrectangular domains and also to nonsymmetric problems [12]. Another robust automatic method is presented in [34].

*Received by the editors November 23, 1993; accepted for publication (in revised form) October 25, 1994.

†Computer Science Department, Technion, Haifa 32000, Israel (yairs@cs.technion.ac.il, israeli@cs.technion.ac.il, sidi@uconnvm.uconn.edu).

None of these methods, however, handles highly indefinite equations; they use coarse-grid operators which are derived from a Galerkin approach, resulting in highly indefinite coarse-grid equations.

All the automatic multilevel methods mentioned above suffer from the disadvantage that for d-dimensional PDE discretizations the coarse-grid operators involve stencils with 3^d coefficients, even when the original difference equation has a $(2d + 1)$-coefficient stencil. This significantly enlarges the amount of arithmetic operations and storage required to generate and store coarse-grid operators (in comparison to algorithms which use $(2d + 1)$-coefficient stencils at all levels). Moreover, it enlarges the cost of a multigrid V-cycle (implemented, e.g., with a Gauss–Seidel smoother) by roughly 25% and 40% for 2-d and 3-d problems, respectively. When W-cycles or more expensive smoothers are used the overhead may be even larger; in particular, the red-black Gauss–Seidel relaxation is not applicable any more and 2^d-color relaxation (whose parallel and vector implementations are more complicated) must be used instead. Furthermore, since the coarse coefficienf matrices lack property A, most of the analysis of [33] for the successive over-relaxation (SOR) method does not apply (the SOR smoother in multigrid methods is considered in [32]). These difficulties are partially relaxed in the algorithm of [31], where $(2^{d+1} - 1)$-coefficient coarse-grid stencils are used; this version, however, is not satisfactory for nonsymmetric problems and problems with discontinuous coefficients.

The aim of this work is to present an automatic multigrid method which does not suffer from the above difficulties; that is, it uses $(2d + 1)$-coefficient stencils only and can be used to solve indefinite problems and several other important classes of problems, e.g., nonsymmetric problems and problems with discontinuous coefficients and nonrectangular domains. Moreover, coarse-grid, restriction, and prolongation operators are obtained from the linear system of equations by a simple and inexpensive recursive process; actually, the cost of this recursion is approximately one work unit, that is, it is equivalent to one fine-grid Gauss–Seidel sweep (compared to five work units for the method of [31]). This fact is especially important for implicit time marching in evolution problems with differential operator or boundary varying in time, where coarse-grid operators are to be reconstructed at every time level. In addition, the fact that operators on different levels are of the same stencil allows easy programming, with data structures and smoothing procedures for coarse-grid equations similar to those used for the finest-grid equation. Like the methods of [11] and [34], the algorithm is robust with respect to the number of fine-grid points, boundary condition type, and shape of the domain. Unlike these methods, however, it is not applicable to schemes which use 3^d-coefficient stencils on the finest grid. We call this algorithm AutoMUG (automatic multigrid).

Our numerical experiments show that, for some difficult problems, the basic AutoMUG iteration may be efficiently accelerated by a Lanczos-type method. It is likely that the existing automatic multigrid methods can also profit from such techniques; indeed, it is shown in [22] that even for highly indefinite, nearly singular Helmholtz equations the two-level implementation of both modified black-box multigrid and AutoMUG can be efficiently accelerated.

AutoMUG is described in §2. In §3 numerical examples are presented. In §4 the algorithm and the numerical results are discussed.

2. The AutoMUG (automatic multigrid) method.

In this section we define the AutoMUG method for the solution of finite difference equations which have $(2d + 1)$-coefficient stencils (for $d = 1, 2$) and examine the properties of its coarse-grid coefficient matrices.

2.1. Abstract definition of AutoMUG.

We begin with an abstract definition of AutoMUG for the solution of the linear system of equations

$$Ax = b.$$

The notation of this definition will be useful in the sequel. In the following, "←" means replacement, S_1 and S_2 are some smoothing procedures, and ϵ, ν_1, ν_2, and o are nonnegative integers denoting, respectively, the cycle index, the number of presmoothings, the number of postsmoothings, and the minimal order of A for which AutoMUG is called recursively. The operators R, P, and Q will be defined later.

```
AutoMUG($x_{\text{in}}, A, b, x_{\text{out}}$) :
        if order($A$) $< o$
                $x_{\text{out}} \leftarrow A^{-1}b$
            otherwise:
                $x_{\text{in}} \leftarrow S_1 x_{\text{in}}$   (repeat $\nu_1$ times)
(1)             $e \leftarrow 0$
                  AutoMUG($e, Q, R(Ax_{\text{in}} - b), e_{\text{out}}$)  }  repeat $\epsilon$ times
                  $e \leftarrow e_{\text{out}}$                           }
                $x_{\text{out}} \leftarrow x_{\text{in}} - Pe$
                $x_{\text{out}} \leftarrow S_2 x_{\text{out}}$   (repeat $\nu_2$ times).
```

For simplicity, we assume in this paper that the method is implemented with a V-cycle ($\epsilon = 1$) and a maximal number of levels ($o = 2$). An iterative application of AutoMUG is given by

```
                $x_0 = 0,\ k = 0$
                while $\|Ax_k - b\|_2 \geq$ threshold $\cdot\ \|Ax_0 - b\|_2$
(2)             AutoMUG($x_k, A, b, x_{k+1}$)
                $k \leftarrow k + 1$
                endwhile
```

Below we define the operators R, P, and Q of (1) for the case in which A is a tridiagonal matrix. Although this is of little significance in itself, it is crucial in the development of the algorithm for other more complicated cases that arise in practical applications and are treated next in this paper.

2.2. The tridiagonal case. In this subsection we define the operators R, P, and Q used in (1) for linear systems which arise, for example, from finite-difference discretization of the ODE

$$(au')' + cu' + \beta u = f \tag{3}$$

where a, c, β, u, and f are functions defined on $\Omega \subset R$ and suitable boundary conditions are imposed on $\partial\Omega$ (this illustrative example will be used in the sequel).

For any matrix B, $B = (b_{i,j})_{1 \leq i \leq K,\ 1 \leq j \leq L}$, define

$$\text{rowsum}(B) \equiv \text{diag}\left(\sum_{j=1}^{L} b_{i,j}\right)_{1 \leq i \leq K}.$$

For a diagonal matrix B, $B = \text{diag}(b_i)_{1 \leq i \leq K}$, let

$$\text{even}(B) \equiv \text{diag}(b_{2i})_{1 \leq i \leq \lfloor K/2 \rfloor} \text{ and } \text{odd}(B) \equiv \text{diag}(b_{2i-1})_{1 \leq i \leq \lceil K/2 \rceil}.$$

Let I denote an identity matrix of a suitable order. Let N be a positive integer, $n = \lfloor \log_2 N \rfloor$, $h \equiv 1/(N+1)$ be the mesh size, and A be a tridiagonal matrix of order N resulting, for example, from a difference scheme approximating (3). For any positive integer K, let $M(K)$ be the permutation matrix which reorders the variables of a K-dimensional vector such that odd-numbered variables appear in the first block and even-numbered variables appear in the second block. Define

$$A_0 = A \text{ and } M_i = M(N/2^i), \quad 0 \le i \le n, \tag{4}$$

where, here and in the sequel, $N/2^i$ means an integer division, that is, $\lfloor N/2^i \rfloor$. In the following we give some motivation for the definition of the operators R, P, and Q used in (1). Suppose A has no vanishing diagonal element and let $D = \text{diag}(A)$. Then, for some bidiagonal matrices B and C, we have

$$A = DM_0^T \begin{pmatrix} I & -B \\ -C & I \end{pmatrix} M_0 = M_0^T(M_0DM_0^T) \begin{pmatrix} I & -B \\ -C & I \end{pmatrix} M_0 = R^{-1}QP^{-1},$$

where

$$R = (M_0DM_0^T) \begin{pmatrix} I & 0 \\ C & I \end{pmatrix} (M_0DM_0^T)^{-1}M_0, \quad Q = (M_0DM_0^T) \begin{pmatrix} I & 0 \\ 0 & I - CB \end{pmatrix},$$

$$P = M_0^T \begin{pmatrix} I & B \\ 0 & I \end{pmatrix}.$$

Suppose these operators are used in (1). Since Q is tridiagonal, one may repeat the above procedure with A replaced by Q to generate suitable operators for the recursive call in (1). With $\nu_1 = \nu_2 = 0$ in (1) this yields a direct solver which is equivalent to the cyclic reduction method [2].

Suppose odd-numbered variables are red-colored. If $\nu_1 = 0$, $\nu_2 = 1$, and the smoothing procedure S_2 of (1) corresponds to the red leg of a red-black Gauss-Seidel relaxation, then an equivalent algorithm is obtained when the definition of R, Q, and P is modified to read

$$\tilde{C} = \text{even}(D)C\text{odd}(D)^{-1}, \ R = \begin{pmatrix} \tilde{C} & I \end{pmatrix} M_0, \ Q = \text{even}(D)(I - CB), \ P = M_0^T \begin{pmatrix} B \\ I \end{pmatrix}.$$

Moreover, $Q = RAP$ still holds; hence this is an appropriate choice for the operators in (1). This procedure is equivalent to that used in [26] for tridiagonal systems.

Note that Q is the Schur complement of A relative to the even-numbered variables. These variables may be viewed as abstract coarse-grid points. Then Q is a coarse-grid operator, R is a fine-to-coarse-grid restriction, and P is a coarse-to-fine-grid prolongation.

We come now to a precise definition of the operators R, P, and Q used in (1). For $0 \le i < n$, define the matrices D_i, B_i, C_i, $P_{A,i+1}$, $\tilde{C}_i$, $R_{A,i+1}$, A_{i+1}, and $S_{A,i+1}$, in this order, by

$$\begin{aligned}
D_i &= \text{diag}(A_i), \\
A_i &= D_iM_i^T \begin{pmatrix} I & -B_i \\ -C_i & I \end{pmatrix} M_i, \\
P_{A,i+1} &= M_i^T \begin{pmatrix} B_i \\ I \end{pmatrix}, \\
\tilde{C}_i &= \text{even}(D_i)C_i\text{odd}(D_i)^{-1}, \\
R_{A,i+1} &= \begin{pmatrix} \tilde{C}_i & I \end{pmatrix} M_i, \\
A_{i+1} &= I - C_iB_i, \\
S_{A,i+1} &= \text{rowsum}(2I - A_{i+1}), \\
A_{i+1} &\leftarrow \text{even}(D_i)A_{i+1}.
\end{aligned} \tag{5}$$

Note that all the A_i are tridiagonal, hence B_i and C_i are well defined (vanishing of a diagonal element of D_i, for some i, is discussed at the end of §2.3). The operator $S_{A,i+1}$ is not used in the present case, but will be helpful in §2.4, where it serves as an approximation to $R_{A,i+1}P_{A,i+1}$. Indeed, when D_0 is a multiple of the identity,

$$\begin{aligned} S_{A,1} - R_{A,1}P_{A,1} &= \text{rowsum}(I + C_0B_0) - (I + C_0B_0) \\ &= \begin{pmatrix} -C_0 & \text{rowsum}(C_0B_0) \end{pmatrix} \begin{pmatrix} B_0 \\ I \end{pmatrix} \sim \frac{h^2}{|a|}\left(\frac{d}{dx}a\frac{d}{dx} + c\frac{d}{dx}\right), \end{aligned} \tag{6}$$

which is negligible when operating on functions u for which $(au')'$ and cu' are not too large. This condition is fulfilled for the solution of the ODE (3), according to the assumptions made in [1].

Remark. The above definition of $S_{A,i+1}$ may be replaced by $S_{A,i+1} = \text{rowsum}(H_i)$, with

$$H_i = R_{A,i+1}P_{A,i+1},\ H_i = R_{A,i+1} \text{ or } H_i = \begin{pmatrix} C_i & I \end{pmatrix}.$$

This yields no essential change in either the theory or the numerical results presented in this paper. The third version is slightly better for indefinite problems, and is used in [23], [24], and [25].

The AutoMUG procedure, namely

$$\text{AutoMUG}(x_{\text{in}}, A, b, x_{\text{out}}),$$

defined in (1), is called $n+1$ times per iteration. In the $(n+1)$st time the AutoMUG procedure is a direct solver. For $1 \le i \le n$, the ith call to the AutoMUG procedure is accomplished with the operators

$$Q \leftarrow A_i,\ P \leftarrow P_{A,i},\ R \leftarrow R_{A,i}.$$

2.3. Properties of the coarse-grid operators. In this subsection we show that the coarse-grid operators A_i defined in (5) preserve some desirable properties of A. To this end, we prove the following lemmas.

For any matrix T let $T^{(i)}$ and $T^{[j]}$ denote its ith row and jth column, respectively, $|T|$ denote the matrix defined by $|T|_{i,j} = |T_{i,j}|$, and $T \ge 0$ (T is nonnegative) hold if all the elements of T are positive or zero. For any two matrices T and S of the same size let $T \ge S$ hold if $T - S \ge 0$. Let $(\cdot,\cdot)$ denote the l_2 inner product and e be the vector whose components are all equal to 1.

LEMMA 2.1. *Assume T and S are two matrices for which the number of rows of S is equal to the number of columns of T. Then*

$$\text{rowsum}(|S|) \le I \ \Rightarrow\ \text{rowsum}(|TS|) \le \text{rowsum}(|T|).$$

Furthermore, if $\text{rowsum}(|S|) \le I$ *and for some* i_0 $\text{rowsum}(|S|)_{i_0,i_0} < 1$ *and* $T^{[i_0]} \ne 0$ *then*

$$\exists\ j_0 \ \textit{s.t.}\ \ \text{rowsum}(|TS|)_{j_0,j_0} < \text{rowsum}(|T|)_{j_0,j_0}.$$

Proof. Suppose $T \ge 0$ and $S \ge 0$. Then

$$\text{rowsum}(TS)_{i,i} = \left(T^{(i)}, \sum_j S^{[j]}\right) \le (T^{(i)}, e) = \text{rowsum}(T)_{i,i}.$$

For the second part of the lemma, by assumption there exists a j_0 such that $T_{j_0,i_0} > 0$. Consequently,

$$\text{rowsum}(TS)_{j_0,j_0} = (T^{(j_0)}, \sum_j S^{[j]}) < (T^{(j_0)}, e) = \text{rowsum}(T)_{j_0,j_0}.$$

For the general case, the lemma follows from

$$\text{rowsum}(|TS|) \le \text{rowsum}(|T||S|).$$

This completes the proof of the lemma.

LEMMA 2.2. *Let I_m be the identity matrix of order m. For some positive integers k and l let W be a matrix of the form*

$$W = \begin{pmatrix} I_k & -S \\ -T & I_l \end{pmatrix},$$

where either T and S are bidiagonal matrices satisfying $T_{i,j} = 0 \Leftrightarrow S_{j,i} = 0$ and $|k - l| \le 1$ or T and S are nonnegative matrices. Then if W is irreducible, so is $I_l - TS$.

Proof. For any square matrix U of order m let $G[U]$ be the directed graph defined by

$$G[U] \equiv \{1, 2, \dots, m\},\ \{(i, j) \mid U_{j,i} \ne 0\}$$

where the first set denotes the nodes and the second one the edges of the graph. Irreducibility of U is equivalent to strong connectivity of $G[U]$, which is equivalent to strong connectivity of $G[I - U]$ (see [30]). From the assumptions of the lemma, there follows, at least for $i \ne j$,

$$((I_{k+l} - W)^2)_{j,i} \ne 0 \ \Leftrightarrow\ \{\exists\, m,\ (I_{k+l} - W)_{j,m} \ne 0,\ (I_{k+l} - W)_{m,i} \ne 0\}.$$

Consequently,

$$\{1, 2, \dots, k + l\},\ \{(i, j) \mid i \ne j \text{ and } \exists\, m \text{ s.t. } (I_{k+l} - W)_{j,m} \ne 0, (I_{k+l} - W)_{m,i} \ne 0\}$$
$$\subset G[(I_{k+l} - W)^2].$$

Let i_1 and i_2 be some integers satisfying $k + 1 \le i_1, i_2 \le k + l$. Since $G[I_{k+l} - W]$ is strongly connected, there exists a path in $G[I_{k+l} - W]$ leading from i_1 to i_2. From the structure of $I_{k+l} - W$ this path must include an even number of edges. Hence i_1 is connected to i_2 also in $G[(I_{k+l} - W)^2]$. The lemma follows from

$$(I_{k+l} - W)^2 = \begin{pmatrix} ST & 0 \\ 0 & TS \end{pmatrix}.$$

This completes the proof of the lemma.

LEMMA 2.3. *Assume T is an M-matrix and* diag$(T) = I$. *Then $(2I - T)T$ is an M-matrix.*

Proof. From the assumptions of the lemma, we have $I - T \ge 0$, and, by Theorem 3.10 of [30], $\rho(I - T) < 1$, where ρ denotes the spectral radius. Hence $(I - T)^2 \ge 0$ and $\rho((I - T)^2) < 1$. Since

$$(2I - T)T = I - (I - T)^2$$

the lemma follows from Theorem 3.8 of [30]. This completes the proof of the lemma.

THEOREM 2.4. *Assume A is tridiagonal and one of the following*: (a) *an M-matrix,* (b) *a nonsingular weakly diagonally dominant matrix, or* (c) *an irreducibly diagonally dominant matrix. Then so are all the matrices A_{i+1} defined in* (4) *and* (5); *moreover, the matrices D_i defined in* (5) *are symmetric positive definite* (*SPD*) *and, in case* (a), *so are the matrices $S_{A,i+1}$.*

Proof. The proof is by induction on i in (5). For case (a), the SPD property of D_i follows from A_i being an M-matrix, and the M-matrix property of A_{i+1} follows from Lemma 2.3 and

$$(2I - D_i^{-1}A_i)D_i^{-1}A_i = M_i^T \begin{pmatrix} I & B_i \\ C_i & I \end{pmatrix} \begin{pmatrix} I & -B_i \\ -C_i & I \end{pmatrix} M_i$$
$$= M_i^T \begin{pmatrix} I - B_iC_i & 0 \\ 0 & I - C_iB_i \end{pmatrix} M_i.$$

For case (b), the SPD property of D_i follows from the nonsingularity and weak diagonal dominance of A_i, the nonsingularity of A_{i+1} follows from

$$D_i^{-1}A_i = M_i^T \begin{pmatrix} I & -B_i \\ -C_i & I \end{pmatrix} M_i$$
$$= M_i^T \begin{pmatrix} I & 0 \\ -C_i & I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & I - C_iB_i \end{pmatrix} \begin{pmatrix} I & -B_i \\ 0 & I \end{pmatrix} M_i,$$

and the weak diagonal dominance of A_{i+1} follows from Lemma 2.1. For case (c), the SPD property of D_i follows from irreducible diagonal dominance of A_i, and the irreducible diagonal dominance of A_{i+1} follows from Lemmas 2.1 and 2.2. The last part of the theorem follows from A_i being an M-matrix. This completes the proof of the theorem.

When its assumption holds, Theorem 2.4 ensures that the matrices B_i and C_i of (5) are well defined and that no division by zero occurs. Otherwise, a diagonal element of D_i (for some i) may vanish; this may be handled by a reasonable definition of the corresponding line of C_i and column of B_i, and, similarly, of the off-diagonal elements in the corresponding line of $R_{A,i+1}$ and column of $P_{A,i+1}$ (e.g., the templates $[0, 1, 0]$ for restriction and $[0, 1, 0]^t$ for prolongation).

2.4. The two-dimensional case. In this subsection we define the operators R, P, and Q used in (1) for linear systems which arise, for example, from finite difference discretization of the PDE

$$(au_x)_x + (bu_y)_y + cu_x + du_y + \beta u = f,$$

where a, b, c, d, β, u, and f are functions defined on $\Omega \subset R^2$ and suitable boundary conditions are imposed on $\partial\Omega$.

Let $U(K, L)$ be a permutation matrix such that for any vector v defined on a $K \times L$ grid and ordered lexicographically row by row, $U(K, L)v$ is the same vector v ordered lexicographically column by column. Let

$$U_{k,l} \equiv U(N/2^k, N/2^l), \qquad 0 \leq k, l \leq n,$$

where, here and in the sequel, $N/2^j$ means an integer division, that is, $\lfloor N/2^j \rfloor$. Suppose A is of the form

$$A = \text{blockdiag}(X^{(j)})_{1 \leq j \leq N} + U_{0,0}^T \cdot \text{blockdiag}(Y^{(j)})_{1 \leq j \leq N} U_{0,0} \tag{7}$$

where $X^{(j)}$ and $Y^{(j)}$ are tridiagonal matrices of order N. For example, if

$$X^{(j)} = Y^{(j)} = \text{tridiag}\left(\frac{1}{h^2}, \frac{-2}{h^2} + \frac{\beta}{2}, \frac{1}{h^2}\right), \quad 1 \leq j \leq N$$

for some $\beta \in C$, then A represents a central discretization of the Helmholtz equation

$$u_{xx} + u_{yy} + \beta u = f$$

in the unit square with Dirichlet boundary conditions.

In the following we give some motivation to the definition of the operators R, P, and Q used in (1). For simplicity, we treat semi-coarsening in the x-direction only; the y-direction coarsening is implemented analogously.

By replacing A in (4) with $X^{(j)}$ and applying (5) to it one may define

$$R_x = \text{blockdiag}(R_{X^{(j)},1})_{1\le j\le N}, \quad P_x = \text{blockdiag}(P_{X^{(j)},1})_{1\le j\le N}.$$

The natural definition $Q = R_x A P_x$ is undesirable because it spoils the tridiagonal structure of the second term in the right-hand side of (7). In order to avoid this, only the first term in the right-hand side of (7) is multiplied by R_x and P_x from the left and right sides, respectively; the second term is multiplied instead by $\text{rowsum}(R_x P_x)$. According to (6), the relative error inserted into $R_x A P_x$ by this approximation is negligible, at least for functions in the neighborhood of the solution.

In order to perform a y-direction coarsening, the second term in the right-hand side of (7) is treated similarly (using restriction and prolongation operators R_y and P_y), while not spoiling the structure of the first term. (1) is then implemented with the resulting coarse-grid operator Q and the restriction and prolongation operators $R = R_y R_x$ and $P = P_x P_y$.

We come now to a precise definition of the operators R, P, and Q of (1) for the 2-d case. Define

$$X_0^{(j)} \equiv X^{(j)}, \quad Y_0^{(j)} \equiv Y^{(j)}, \quad 1 \le j \le N.$$

For $i = 1, \ldots, n$, define the matrices R_i, P_i, and A_i, in this order, by

For $1 \le j \le N/2^{i-1}$, do (5) with A and i replaced by $X^{(2^{i-1}j)}$ and $i-1$, respectively,

For $1 \le j \le N/2^{i-1}$, do (5) with A and i replaced by $Y^{(2^{i-1}j)}$ and $i-1$, respectively,

$$R_i \equiv U_{i,i}^T \cdot \text{blockdiag}(R_{Y^{(2^i j)},i})_{1\le j\le N/2^i} U_{i-1,i} \cdot \text{blockdiag}(R_{X^{(2^{i-1} j)},i})_{1\le j\le N/2^{i-1}},$$

$$P_i \equiv \text{blockdiag}(P_{X^{(2^{i-1} j)},i})_{1\le j\le N/2^{i-1}} U_{i-1,i}^T \cdot \text{blockdiag}(P_{Y^{(2^i j)},i})_{1\le j\le N/2^i} U_{i,i},$$

$$\text{blockdiag}(X_i^{(2^i j)})_{1\le j\le N/2^i} \quad \leftarrow$$

$$\leftarrow U_{i,i}^T \cdot \text{blockdiag}(S_{Y^{(2^i j)},i}))_{1\le j\le N/2^i} U_{i,i} \cdot \text{blockdiag}(X_i^{(2^i j)})_{1\le j\le N/2^i}, \tag{8}$$

$$\text{blockdiag}(Y_i^{(2^i j)})_{1\le j\le N/2^i} \quad \leftarrow$$

$$\leftarrow U_{i,i} \cdot \text{blockdiag}(S_{X^{(2^i j)},i}))_{1\le j\le N/2^i} U_{i,i}^T \cdot \text{blockdiag}(Y_i^{(2^i j)})_{1\le j\le N/2^i},$$

$$A_i \equiv \text{blockdiag}(X_i^{(2^i j)})_{1\le j\le N/2^i} + U_{i,i}^T \cdot \text{blockdiag}(Y_i^{(2^i j)})_{1\le j\le N/2^i} \cdot U_{i,i}.$$

As in the tridiagonal case, the ith call to the AutoMUG procedure in (1), $1 \le i \le n$, is accomplished with the operators

$$Q \leftarrow A_i, \quad P \leftarrow P_i, \quad R \leftarrow R_i.$$

The following theorem ensures that the coarse-grid operators A_i enjoy a desirable property.

THEOREM 2.5. *Assume A is defined as in* (7) *with $X^{(j)}$ and $Y^{(j)}$ being tridiagonal irreducibly diagonally dominant M-matrices of order N. Then all the matrices A_i defined in* (8) *are irreducibly diagonally dominant M-matrices.*

Proof. Following the route of the proof of Theorem 2.4, by induction on i in (8) all the tridiagonal matrices X_i and Y_i are irreducibly diagonally dominant M-matrices and the diagonal matrices S_i are SPD. Hence the A_i are weakly diagonally dominant, with a strict diagonal dominance for at least one row. Irreducibility of A_i follows from

$$G[A_i] = \{1, 2, \dots, N/2^i\} \times \{1, 2, \dots, N/2^i\},$$

$$\{((k,l),(m,l)) \mid (X_i^{(2^i l)})_{m,k} \neq 0\} \cup \{((l,k),(l,m)) \mid (Y_i^{(2^i l)})_{m,k} \neq 0\}.$$

Hence the A_i are irreducibly diagonally dominant matrices. Since the A_i have positive diagonal elements and nonpositive off-diagonal elements, it follows from [30] that they are also M-matrices. This completes the proof of the theorem.

The definition (7) of the coefficient matrix A assumes that the grid is logically rectangular. Problems involving complicated domains may be treated by embedding the original grid into a logically rectangular grid [28] and appending trivial equations for the redundant variables. This approach was used in the numerical experiments in §3. Note that these dummy variables, as well as their corresponding equations, do not have to be stored, since they do not influence the values of the original variables.

3. Numerical experiments. Our aim in this section is to show the applicability of AutoMUG to several classes of problems, including indefinite Helmholtz equations. We compare the performance of AutoMUG to that of a standard multigrid version which has the same complexity, that is, uses $(2d + 1)$-coefficient coarse-grid stencils only (see §1). This version, denoted by MG, is implemented as follows: coarse-grid operators are generated from the PDE by the same scheme as that used on the finest grid. Full weighting and bilinear interpolation are used for restriction and prolongation, respectively. For both AutoMUG and MG the maximal number of levels (e.g., six levels for 63 × 63 grids) is used, which means setting $o = 2$ in (1); the only exceptions are examples 13 and 14, where four levels are used.

When implementing MG one must use $2^{n+1} - 1$ grid points on the finest grid and $2^q - 1$, $1 \leq q \leq n$, for coarser grids in order to preserve uniformity. Here the even points, which are taken as coarse-grid points, are always internal points of the original grid. For 2^n-point grids, on the other hand, the last fine-grid point appears as a last grid point in all grids. Hence, coarse grids are biased towards the boundary. For AutoMUG, on the other hand, grids of both 2^n points or $2^{n+1} - 1$ points may be used, and actually achieve the same convergence rates; this is because in the 2^n points case AutoMUG automatically chooses in the coarse-grid schemes the most accurate extrapolation of boundary points. An odd number of points $N = 63$ is used here for an easier comparison between AutoMUG and MG.

On all grids the smoother was either the one provided by the ILU(1,1) decomposition of [18] and [29] (namely, ILU with no fill-in) or the red-black Gauss-Seidel (RB) relaxation. ILU was considered as a smoother in multigrid in [16] and 17]. One presmoothing and one postsmoothing is performed in each level of a V-cycle ($\epsilon = \nu_1 = \nu_2 = 1$ in (1)). The initial guess is random in (0, 1). Double-precision arithmetic is used.

AutoMUG and MG are iterated according to (2) with $10^{-12} \leq$ threshold $\leq 10^{-14}$, to avoid the effect of round-off errors. It was checked that the l_∞ norm of the error is reduced during this process by more than 10 orders of magnitude. Convergence factors are computed from the last four iterations, namely

$$\text{convergence factor} = \left(\frac{\|Ax_{\text{last}} - b\|_2}{\|Ax_{\text{last}-3} - b\|_2} \right)^{1/3}$$

where *last* is the index of the last iteration. When the basic iteration (2) by itself diverges (denoted by "*") or unsatisfactorily converges, it is also implemented with an acceleration

TABLE 1

Convergence factors for MG and AutoMUG (in parentheses: averaged convergence factors when the TFQMR acceleration is used).

	MG	MG	AutoMUG	AutoMUG
example	ILU	RB	ILU	RB
1	0.067	0.102	0.068	0.096
2a	0.474 (0.148)	0.96 (0.667)	0.474 (0.148)	0.96 (0.667)
2b	0.005	0.99 (0.83)	0.005	0.99 (0.83)
3	0.127	0.58 (0.330)	0.127	0.58 (0.330)
4	0.181	0.287	0.088	0.198
5a	0.235	0.49 (0.164)	0.114	0.199
5b	0.208	0.44 (0.159)	0.112	0.198
6	0.196	0.574 (0.269)	0.154	0.424 (0.171)
7	* (0.525)	*	0.162	0.427 (0.192)
8	0.573 (0.159)	0.886 (0.46)	0.220	0.526 (0.215)
9	* (0.57)	*	0.196	0.526 (0.205)
10a	0.057	0.458 (0.427)	0.063	0.148
10b	0.26	*	0.120	*
11	*	*	0.691 (0.243)	0.865 (0.454)
12	0.53 (0.246)	0.79 (0.414)	0.994 (0.183)	* (0.392)
13	*	* (0.60)	* (0.70)	0.95 (0.314)
14	*	* (0.714)	* (0.543)	0.85 (0.367)

method applied to it. We have used the Transpose-free quasi-minimal residual (TFQMR) method (Algorithm 5.2 in [15]) which avoids the computation of the transpose of the coefficient matrix and preconditioner (the latter is only implicitly given in (1), so its transpose is not available). As a matter of fact, the TFQMR method may be considered a modification of the conjugate gradients squared (CGS) method of [27] and [28], which is a generalization of the conjugate gradients (CG) method to nonsymmetric and indefinite problems. We have found that the performance of CGS is similar to that of TFQMR; however, we have preferred the latter because of its smooth convergence curve.

All the above acceleration techniques require an amount of storage and arithmetical operations comparable to that of CG, namely an additional 1 – 1.5 work units per iteration. For the accelerated iteration the convergence factor defined above often oscillates; hence the averaged convergence factor defined by

$$\text{averaged convergence factor} = \left(\frac{\|Ax_{\text{last}} - b\|_2}{\|Ax_0 - b\|_2}\right)^{1/\text{last}}$$

is considered instead, and displayed in parentheses in Table 1. When the accelerated iteration stagnates, the sign "$*$" alone is presented.

The problems solved are of the form

$$Lu(x, y) = f(x, y), \quad (x, y) \in \Omega \subset R^2,$$

with the exact solution $u = xy$ (except of examples 7, 9, 10b, and 12, for which the exact solution is $u = 0$). Since the initial guess is random and the problem is linear, the rates of convergence are independent of the specific choice of the solution. A second-order central finite difference scheme is used. For most examples the region Ω is the unit square, Dirichlet boundary conditions are imposed, and uniform grids are used. Exceptions to the above are noted at particular examples.

It is seen from the numerical results that for some examples AutoMUG by itself diverges or unsatisfactorily converges, while when supplemented with an acceleration scheme it converges quickly; this is apparently because the iteration matrix has some isolated eigenvalues of large magnitude, while most of its spectrum is clustered around zero.

List of examples.

1. The Poisson equation $L = -\Delta$.
2. The anisotropic equation

$$-u_{xx} - \sigma u_{yy} = 0$$

with (a) $\sigma = 10^{-2}$ and (b) $\sigma = 10^{-4}$. This example is more difficult than the Poisson equation, since error components which are smooth in the x-direction and oscillate in the y-direction are not easily smoothed by a point Gauss-Seidel smoother; this difficulty may be handled by employing an appropriate line relaxation [4].

The ILU smoother used here is lexicographically ordered; hence oscillations in the x-direction are smoothed much better than those of the y-direction. According to the above remark, it is likely that an anti-lexicographical ordering is more suitable to this example.

Case (a) is the most difficult one considered in [31]. The convergence factors derived there are similar to those obtained here.

3. The Poisson equation with a Chebyshev-type grid; it is discretized via central differences on the two-dimensional grid

$$P_N(j,k) \equiv \left(\frac{1-\cos\left(\dfrac{j\pi}{N+1}\right)}{2}, \frac{1-\cos\left(\dfrac{k\pi}{N+1}\right)}{2} \right), \quad 1 \le j,k \le N.$$

The matrix operator for this scheme may be used as a preconditioner for a Chebyshev-collocation discretization of the Poisson equation (see [19]). Coarse-grid operators for MG are obtained from the discretization of the PDE on the grids $P_{\lfloor N/2^i \rfloor}$, $1 \le i \le 5$. As in the previous example, the RB smoother is inferior to ILU; the reason for this is that the equation is locally anisotropic in most of the mesh cells.

4. The Poisson equation in a square with a slit. The actual shape is a 63 × 63 grid minus a narrow strip of width 2 points and length 32 points emerging from the center in the x-direction. This problem is considered more difficult than the Poisson equation in a square (see [5]).
5. The Poisson and definite Helmholtz equations in a domain approximating the North Atlantic, from about 10° to 53° north. A definite Helmholtz problem of the form

$$-u_{xx} - u_{yy} + \beta u = f$$

stems from explicit time-stepping in the quasi-geostrophic version of the shallow-water equations. The domain is embedded in a Cartesian grid of 45 × 93 points. The bounding points are assumed to lie on that grid. The cases under consideration are (a) $\beta = 0$ and (b) $\beta = 20$.

6. The Poisson equation with boundary conditions of the third kind

$$\frac{\partial u}{\partial n} + 1.5u = 0,$$

where $\vec{n}$ denotes the outer normal vector. This example is presented mainly for the sake of comparison with the next one.

For this example MG was implemented with restriction and prolongation operators modified near the boundaries such that their rowsums equal 1 everywhere; otherwise, much slower convergence occurs. For all other examples, however, MG with this modification was inferior to MG with the standard bilinear interpolation and full-weighting restriction.

7. The diffusion equation

$$-(a(x,y)u_x)_x - (a(x,y)u_y)_y = 0$$

with the discontinuous coefficient

$$a(x,y) = \begin{cases} \gamma & \max(|x-0.5|, |y-0.5|) \le 3h, \\ 1 & \text{otherwise,} \end{cases}$$

where h is the mesh size and $\gamma = 10^4$ and the boundary conditions of the previous example. It is known [1] that standard multigrid approaches cannot simulate the problem appropriately on coarse grids.

This problem is similar to Problem 3 in [34] and to the most difficult case of Problem 1 in [1]. The fine-grid (for MG, also the coarse-grid) discretization is done as in [1].

8. The Poisson equation on the L-shaped region

$$\Omega = ((0,1) \times (0,0.5)) \cup ((0,0.5) \times (0.5,1)).$$

Dirichlet and Neumann boundary conditions are imposed on

$$\Gamma \equiv (\{0.5\} \times [0.5,1]) \cup ([0.5,1] \times \{0.5\})$$

and $\partial\Omega \setminus \Gamma$, respectively. This example is presented mainly for the sake of comparison with the next one.

9. A diffusion equation in the spirit of Kershaw's problem (Problem 7 in [34]) is solved in the same L-shaped region as above. The equation is that of example 7 with $\gamma = 10^6$. Boundary conditions of the same type as those of example 8 are imposed.

By comparing the results of examples 7 and 9 to those of examples 6 and 8 (respectively) it may be concluded that the discontinuity inserted in examples 7 and 9 does not affect the efficiency of AutoMUG.

10. The convection-diffusion equation with fan-like streamlines

$$-u_{xx} - u_{yy} + \eta(xu_x + yu_y) = f$$

whose characteristics are rays starting at the origin so that they all intersect a boundary. This kind of equation is hard to solve with the multigrid approach, since error terms which are smooth in the convection direction and oscillate in the perpendicular direction are only half-corrected by the coarse-grid term [9].

Two cases were examined: (a) $\eta = 150$, for which diagonal dominance holds and (b) $\eta = 300$, for which it is violated for most of the equations in the linear system.

Unlike all other examples in this section, MG was implemented with coarse-grid operators derived from an upwind, rather than central, scheme; otherwise, considerably slower convergence (or even divergence) was reported.

11. The circulating flow equation

$$\sin(\pi(y-0.5))\cos(\pi(x-0.5))u_x - \sin(\pi(x-0.5))\cos(\pi(y-0.5))u_y = f.$$

The region is a square with a 1 × 1 point hole at the middle of it. For this region, an upwind scheme is inadequate [8]; following [10], we have thus inserted isotropic artificial viscosity, the amount of which is locally chosen to be the minimal amount required for weak diagonal dominance. The results for AutoMUG are far better than those of the V-cycle in [10]. The coarse-grid operators generated by AutoMUG may thus be used in conjunction with the defect correction approach of [10] to accelerate convergence.

For the MG approach divergence was reported no matter whether coarse-grid operators are derived from the central or upwind scheme.

Convection diffusion equations similar to those considered here are solved efficiently in [34]. The method of [34], however, uses the incomplete line LU (ILLU) smoother; this is a robust smoother, which also achieves high rates of convergence when used as a preconditioner in preconditioned CG (with no multigrid strategy) [28]. Since we are interested in investigating the efficiency of multigrid methods on their own, we avoid implementing AutoMUG with smoothers which are also efficient preconditioners.

The last three examples are of special difficulty, involving oscillating coefficients or indefiniteness.

12. The diffusion equation

$$-(a(x-y)u_x)_x - (a(x-y)u_y)_y = 0$$

with the oscillating coefficient

$$a(t) = 1.05 + \sin\left(\frac{t}{\sqrt{2}h}\right)$$

(see [13]). The discretization is symmetric, as in [30].

13. The indefinite Helmholtz equation

$$-u_{xx} - u_{yy} - \beta u = f$$

with $\beta = 200$. Note that for $h = 1/64$ the (k, l) eigenvalue of the Poisson equation, with $(k, l) \in \{(2, 4), (4, 2)\}$, is equal to

$$\frac{4}{h^2}\left(\sin^2\frac{kh}{2} + \sin^2\frac{lh}{2}\right) = 196.8537;$$

hence, with the above choice for β, the coefficient matrix has nearly singular eigenvalues (see [7]). Furthermore, eight distinct eigenvalues of the Helmholtz equation are negative; hence the problem is indefinite, and the iteration matrix for either AutoMUG or MG often has eigenvalues of magnitude larger than 1 (see [22]-[24]). The use of an acceleration scheme is thus crucial to ensure convergence.

For the current and the following example only four levels are used, and the fourth-level equation is approximately solved by 100 Kacmarz sweeps. The reason for this is that the coefficient matrix for the fifth-level problem is nonpositive for either AutoMUG or MG, hence cannot serve as a suitable approximation to the PDE (see [23] and [24] for a detailed explanation).

14. The indefinite Helmholtz equation (Example 6.3 in [14])

$$-u_{xx} - u_{yy} - \beta u = f$$

with $\beta = 200$, complex boundary conditions of the third kind

$$\frac{\partial u}{\partial n} + 10iu = g, \quad (x, y) \in [0, 1] \times \{0\},$$

and Dirichlet boundary conditions on the rest of $\partial\Omega$. The mixed boundary conditions are discretized via a first-order scheme as in [14]. Like the previous problem, this problem is indefinite, hence the use of acceleration is crucial.

Unlike most of the examples in this section, for the indefinite examples RB is a better smoother than ILU. We believe this is due to the instability of ILU for coarse-grid equations; this may be handled by adding some positive weights to main diagonal elements in the ILU decomposition which become too small.

4. Discussion. AutoMUG is a multilevel method for the solution of finite difference schemes of $(2d+1)$-coefficient stencils which arise, for example, from d-dimensional second-order PDEs. It is automatic in the sense that its definition depends on the scheme on the original grid only, and no rediscretization of the PDE is required. Derivation of coarse-grid, restriction, and prolongation operators for AutoMUG is inexpensive and straightforward. In addition, property A of the coefficient matrix is preserved at all levels; this simplifies the programming and enables the use of the RB and SOR smoothers.

The numerical examples show that, when implemented with a suitable smoother, AutoMUG gives high rates of convergence for several classes of problems: symmetric, nonsymmetric and problems with discontinuous coefficients, nonuniform grids, and nonrectangular domains. When supplemented with an acceleration scheme, high rates of convergence also are achieved for pure convection problems and indefinite Helmholtz equations.

For some problems AutoMUG is inferior to nonautomatic multigrid algorithms designed especially for the specific problem. In particular, it is inferior to the method of [7] for slightly indefinite problems and to that of [10] for problems with circulating flow. In these cases, it is recommended that the coarse-grid operators of AutoMUG be used in conjunction with the specific approach, that is, the projection of [7] or the overresidual weighting and defect correction of [10]. Alternatively, accelerating AutoMUG by a Lanczos-type method also yields high rates of convergence. For highly indefinite problems, the use of such acceleration is crucial, since the basic iteration often diverges. Hence for problems which involve several sources of difficulty, e.g., indefiniteness, convection, jumps, singularities, etc., AutoMUG supplemented with an acceleration scheme seems to provide an effective solver.

Acknowledgment. We wish to thank Dr. Irad Yavneh for valuable comments and for suggesting examples 4 and 5.

REFERENCES

[1] R. Alcouffe, A. Brandt, J. E. Dendy, and J. Painter, *The multigrid method for the diffusion equation with strongly discontinuous doefficients*, SIAM J. Sci. Statist. Comput., 2 (1981), pp. 430-454.

[2] P. Amodio and F. Mazzia, *Backward error analysis of cyclic reduction for the solution of tridiagonal systems*, Math. Comp., 62 (1994), pp. 601-617.

[3] J. H. Bramble, Z. Leyk, and J. E. Pasciak, *Iterative schemes for non-symmetric and indefinite elliptic boundary value problems*, Math. Comp., 60 (1993), pp. 1-22.

[4] A. Brandt, *Multi-level adaptive solutions to boundary-value problems*, Math. Comp., 31 (1977), pp. 333-390.

[5] ———, *Guide to Multigrid Development, Multigrid Methods*, W. Hackbusch and U. Trottenberg, eds., Lecture Notes in Mathematics 960, Springer-Verlag, Berlin, Heidelberg, 1982.

[6] ———, *Algebraic multigrid theory: The symmetric case*, Appl. Math. Comp., 19 (1986), pp. 23-56.

[7] A. Brandt and S. Ta'asan, *Multigrid methods for nearly singular and slightly indefinite problems*, in Multigrid Methods II, Proceedings, Cologne, 1985, Lecture Notes in Mathematics 1228, W. Hackbusch and U. Trottenberg, eds., Springer-Verlag, Berlin, pp. 100-122.

[8] A. Brandt and I. Yavneh, *Inadequacy of first-order upwind difference schemes for some recirculating flows*, J. Comp. Phys., 93 (1991), pp. 128-143.

[9] ———, *On multigrid solution of high Reynolds incompressible entering flows*, J. Comp. Phys., 101, (1992), pp. 151-164.

[10] A. Brandt and I. Yavneh, *Accelerated multigrid convergence and high Reynolds recirculating flows*, SIAM J. Sci. Statist. Comput., 14 (1993), pp. 607-626.

[11] J. E. Dendy, *Black box multigrid*, J. Comp. Phys., 48 (1982), pp. 366-386.

[12] ———, *Black box multigrid for nonsymmetric problems*, Appl. Math. Comp., 13 (1983), pp. 261-283.

[13] B. Engquist and E. Luo, *Multigrid methods for differential equations with highly oscillating coefficients*, Sixth Copper Mountain Conference on Multigrid Methods, N. D. Melson, S. F. McCormick, and T. A. Manteuffel, eds., NASA Langley Research Center, Hampton, VA, 1993, pp. 175-190.

[14] R. W. Freund, *Conjugate gradients type methods for linear systems with complex symmetric coefficient matrices*, SIAM J. Sci. Comput., 13 (1992), pp. 425-448.

[15] ———, *Transpose free quasi-minimal residual algorithm for non-Hermitian linear systems*, SIAM J. Sci. Comput., 14 (1993), pp. 470-482.

[16] R. Kettler, *Analysis and comparison of relaxation schemes in robust multigrid and preconditioned conjugate gradients methods*, in Multigrid Methods, Lecture Notes in Mathematics 960, W. Hackbusch and U. Trottenberg, eds., Springer-Verlag, Berlin, Heidelberg, 1982.

[17] R. Kettler and J. A. Meijerink, *A Multigrid Method and a Combined Multigrid-Conjugate Gradient Method for Elliptic Problems with Strongly Discontinuous Coefficients in General Domains*, Shell Publ. 604, KSELP, Rijswijk, the Netherlands, 1981.

[18] J. A. Meijerink and H. A. Van der Vorst, *Guidelines for the usage of incomplete decompositions in solving sets of linear equations as they occur in practical problems*, J. Comp. Phys., 44 (1981), pp. 134-155.

[19] A. Quarteroni and E. Zampieri, *Finite element preconditionering for Legendre spectral collocation approximations to elliptic equation and systems*, SIAM J. Numer. Anal., 29 (1992), pp. 917-936.

[20] J. W. Ruge and K. Stuben, *Efficient solution of finite difference and finite element equations by algebraic multigrid*, in Multigrid Methods for Integral and Differential Equations, D. J. Paddon and H. Holstein, eds., Oxford Univ. Press., New York, 1985, pp. 169-212.

[21] ———, *Algebraic Multigrid*, in Multigrid Methods, S. F. McCormick, ed., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1987.

[22] Y. Shapira, *Two-Level Analysis of Automatic Multigrid for SPD, Non-Normal and Indefinite Problems*, Tech. Rep. #824, Computer Science Department, Technion–Israel Institute of Technology, July 1994; Numer. Math., submitted.

[23] ———, *Multigrid Techniques for 3-D Definite and Indefinite Problems with Discontinuous Coefficients*, Tech. Rep. #834, Computer Science Department, Technion–Israel Institute of Technology, Oct. 1994; Math. Comp., submitted.

[24] ———, *Multigrid Techniques for Highly Indefinite Equations*, 8th Copper Mountain Conference on Multigrid Methods, S. F. McCormick and T. A. Manteuffel, eds., NASA Langley Research Center, Hampton, VA, 1995, in press.

[25] Y. Shapira, M. Israeli, and A. Sidi, *An automatic multigrid method for the solution of sparse linear systems*, Sixth Copper Mountain Conference on Multigrid Methods, N. D. Melson, S. F. McCormick, and T. A. Manteuffel, eds., NASA Langley Research Center, Hampton, VA, 1993, pp. 567-582.

[26] R. A. Smith and A. Weiser, *Semicoarsening multigrid on a hypercube*, SIAM J. Sci. Comput., 13 (1992), pp. 1314-1329.

[27] P. Sonneveld, *CGS, a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 36-52.

[28] P. Sonneveld, P. Wesseling, and P. M. de Zeeuw, *Multigrid and conjugate gradient methods as convergence acceleration techniques*, in Multigrid Methods for Integral and Differential Equations, D. J. Paddon and H. Holstein, eds., Oxford Univ. Press, U.K., 1985, pp. 117-168.

[29] H. A. Van der Vorst, *Iterative solution methods for certain sparse linear systems with a non-symmetric matrix arising from PDE-problems*, J. Comp. Phys., 49 (1982), pp. 1-19.

[30] R. Varga, *Matrix Iterative Analysis*, Prentice-Hall, NJ, 1962.

[31] P. Wesseling, *A robust and efficient multigrid solver*, Multigrid Methods, W. Hackbusch and U. Trottenberg, eds., Lecture Notes in Mathematics 960, Springer-Verlag, Berlin, Heidelberg, 1982, pp. 614-630.

[32] I. Yavneh, *On Red Black SOR Smoothing in Multigrid*, SIAM J. Sci. Comput., submitted.

[33] D. Young, *Iterative Solution of Large Linear Systems*, Academic Press, NY, 1971.

[34] P. M. de Zeeuw, *Matrix-dependent prolongations and restrictions in a blackbox multigrid solver*, J. Comput. Appl. Math., 33 (1990), pp. 1-27.

SIAM J. SCI. COMPUT.
Vol. 17, No. 2, pp. 454-478, March 1996

010

ALTERNATING-DIRECTION LINE-RELAXATION METHODS ON MULTICOMPUTERS*

JÖRN HOFHAUS† AND ERIC F. VAN DE VELDE‡

Abstract. We study the multicomputer performance of a three-dimensional Navier–Stokes solver based on alternating-direction line-relaxation methods. We compare several multicomputer implementations, each of which combines a particular line-relaxation method and a particular distributed block-tridiagonal solver. In our experiments, the problem size was determined by resolution requirements of the application. As a result, the granularity of the computations of our study is finer than is customary in the performance analysis of concurrent block-tridiagonal solvers. Our best results were obtained with a modified half-Gauss-Seidel line-relaxation method implemented by means of a new iterative block-tridiagonal solver that is developed here. Most computations were performed on the Intel Touchstone Delta, but we also used the Intel Paragon XP/S, the Parsytec SC-256, and the Fujitsu S-600 for comparison.

Key words. Navier–Stokes equations, concurrency, parallelism, block-tridiagonal systems, tridiagonal systems, ADI, alternating directions

AMS subject classifications. 65M20, 65N40, 65Y05, 76C05, 76M20

1. Introduction.

When using alternating-direction line-relaxation methods for systems of partial-differential equations discretized on a rectangular grid, one must solve many block-tridiagonal systems of linear equations in every relaxation step. This type of computation surfaces in many applications. In our work, we faced it when parallelizing a highly vectorized solver for the three-dimensional, unsteady, and incompressible Navier–Stokes equations [4] for use on multicomputers. In this paper, we shall discuss and analyze five line-relaxation methods and six distributed block-tridiagonal solvers used during the course of this project. We have measured the performance of several combinations of relaxation methods and block-tridiagonal solvers on three multicomputers (the Intel Touchstone Delta [5], the Intel Paragon XP/S [6], and the Parsytec SC-256 [13]) and on one conventional vector processor (the Fujitsu S-600).

Three-dimensional computations are more complex than two-dimensional ones because of additional coordinates for the geometry and the vector fields. Three-dimensional computations are also algorithmically different, because every line-relaxation step requires the solution of a larger number of smaller-sized linear systems than a comparable two-dimensional line-relaxation step. To see this, consider a two- and a three-dimensional problem of the same size, i.e., with the same number of unknowns. The two-dimensional problem on an $M \times M$ grid requires the solution of $O(M)$ block-tridiagonal systems of size $O(M)$, while the three-dimensional problem on an $N \times N \times N$ grid with $N^3 = M^2$ requires the solution of $O(N^2) = O(M^{4/3})$ systems of size $O(N) = O(M^{2/3})$.

On multicomputers, the reduced system size has a serious impact. There exist many concurrent algorithms to solve block-tridiagonal systems; see, e.g., [2], [10], [12], [14]–[16]. However, these methods have been studied mainly for very coarse-grain computations, i.e., for computations in which the ratio of the system size over the number of nodes is high. Bondeli [2] studied computations with less than 25 nodes and system sizes exceeding 16, 800. Krechel,

*Received by the editors August 16, 1993; accepted for publication (in revised form) October 24, 1994. This material is based upon work supported by the NSF under Cooperative Agreement No. CCR-9120008. Access to the Intel Touchstone Delta was provided by the Concurrent SuperComputing Consortium.

†Aerodynamisches Institut der RWTH Aachen, Wüllnerstr. zw. 5 u. 7, 52062 Aachen, Germany (jorn@aia.rwth-aachen.de). Caltech visit sponsored by the German Research Association (DFG) within the project: Strömungssimulation mit Hochleistungsrechnern.

‡Applied Mathematics 217-50, California Institute of Technology, Pasadena, CA 91125 (evdv@ama.caltech.edu).

Plum, and Stüben [14] improved the efficiency of a modified cyclic-reduction algorithm on the 16-node iPSC/2-VX. Their computations of moderate-to-coarse granularity solved 256 tridiagonal systems of size 128 and the highest-achieved efficiency was about 50%. For the reasons mentioned above, the granularity of the block-tridiagonal solver in our three-dimensional application is much smaller when running the code for a grid of fixed size on all 512 computing nodes of the Delta. Constructing an efficient concurrent program for this problem is a challenge.

An outline of the paper follows. In §2, a brief description of the implemented relaxation methods is given. In §3, we shall study algorithm and concurrency issues of several basic concurrent alternating-direction relaxation methods. Each method is based on a different distributed solver for block-tridiagonal systems. First, the sequential block-tridiagonal solver is distributed and pipelined to obtain a concurrent line-relaxation method. Subsequently, we use concurrent tridiagonal solvers based on recursive doubling, cyclic reduction, partitioning, and divide and conquer, respectively. Finally, a new iterative tridiagonal solver is developed. In §4, we analyze the performance on the Delta of the proposed methods applied to a Navier-Stokes solver. The Navier-Stokes equations and their discretization for an interesting application are given in §4.1. We give this description for the sole purpose of defining precisely how the performance data were obtained. The fluid-dynamical results of our computations are discussed in another paper [1]. In §5, computations on the Paragon, the Parsytec, and the Fujitsu are compared with those on the Delta.

2. Relaxation methods. The discretization of partial-differential equations often leads to a large sparse system of equations

$$A\vec{u} = \vec{f}. \tag{1}$$

Classical relaxation methods to solve (1) are obtained by splitting the coefficient matrix A into

$$A = G + H$$

with G an easily inverted matrix. This defines the iteration

$$G\vec{u}^{(\nu)} = \vec{f} - H\vec{u}^{(\nu-1)}, \tag{2}$$

which converges to the exact solution $\vec{u}^*$ if the spectral radius of $G^{-1}H$ is less than one; e.g., see [11].

In the description that follows, it is convenient to think of the three-dimensional Poisson problem discretized to second-order accuracy on a three-dimensional rectangular grid of size $M \times N \times K$. Then, interior grid points are identified by means of a triple (m, n, k) with $1 \leq m \leq M$, $1 \leq n \leq N$, and $1 \leq k \leq K$. One unknown and one equation is associated with each interior grid point.

One *elementary line-relaxation step* updates all unknowns of one grid line simultaneously. For example, an x-line is a set of grid points (m, n, k) where $1 \leq m \leq M$ and n and k are fixed. An x-line is, therefore, identified by means of a couple (n, k). An elementary x-line-relaxation step reduces to the tridiagonal system

$$a_m\, u_{m-1} \;+\; b_m\, u_m \;+\; c_m\, u_{m+1} \;=\; d_m, \qquad m = 1, \ldots, M \tag{3}$$

for each x-line. One *x-line-relaxation step* updates all x-lines of the grid once, i.e., it performs an elementary x-line-relaxation step for all (n, k) with $1 \leq n \leq N$ and $1 \leq k \leq K$. For three-dimensional problems, an *alternating-direction line-relaxation step* consists of an *x-line-relaxation step*, a *y-line-relaxation step*, and a *z-line-relaxation step*. We restrict our discussion to the x-line-relaxation step. In matrix-vector notation, the system (3) has the form

$$T\,\vec{u} = \vec{d}, \tag{4}$$

with

$$T = \begin{pmatrix} b_1 & c_1 & & & & \\ a_2 & b_2 & c_2 & & \mathbf{0} & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & \cdot & \cdot & \cdot \\ & \mathbf{0} & & a_{M-1} & b_{M-1} & c_{M-1} \\ & & & & a_M & b_M \end{pmatrix}.$$

The right-hand side vector $\vec{d}$ depends on the boundary conditions, which add the terms $a_1 u_0$ and $c_M u_{M+1}$ to the right-hand side of the first and the last equation, respectively. The right-hand side vector also depends on neighboring x-lines $(n+1, k)$, $(n-1, k)$, $(n, k-1)$, and $(n, k+1)$. (This assumes a classical seven-point stencil for second-order discretization of the Poisson equation in three dimensions.) Three variants of the line-relaxation method were implemented.

When the right-hand side terms for x-line (n, k) are computed exclusively with u-values that were known before the x-line-relaxation step began, the method is called *Jacobi line relaxation.* In this case, all elementary x-line-relaxation steps and all systems (3) are independent. In principle, all tridiagonal solves may be performed concurrently provided that the coefficients of T are not distributed over several processes.

Gauss–Seidel line relaxation assigns an order to the x-lines and uses the updated u-values obtained in preceding elementary x-line-relaxation steps to compute the right-hand sides of the current elementary x-line-relaxation step. *Lexicographic ordering* approach enforces a rigid and sequential ordering on the solution of the tridiagonal systems: (n, k) follows $(n-1, k)$ and $(n, k-1)$.

For the purpose of vectorization and/or concurrent computing, it makes sense to make many tridiagonal systems independent of one another so that they can be solved simultaneously. *Red-black ordering* performs elementary line-relaxation steps first for all x-lines for which $n+k$ is even and, subsequently, for all x-lines for which $n+k$ is odd.

Lexicographic ordering may be kept provided the relaxation method is modified. For the x-line-relaxation step, e.g., the systems of grid plane $k =$ constant are made independent of those of grid plane $k-1$ by using the old u-values of x-line $(n, k-1)$ when computing the system for the u-values for x-line (n, k). In the remainder of this paper, we call this *half-Gauss-Seidel line relaxation.* In principle, grid planes may now be solved concurrently and/or the solution of all tridiagonal systems (n, k) for fixed n may be vectorized. Because the nodes of our target computers contain vector processors, all our implementations use the vectorization in the z-direction for the x-line-relaxation step.

The $M \times N \times K$ grid is distributed over a $P \times Q \times R$ process grid. The data distribution over P processes in the x-direction forces us to use a distributed tridiagonal solver for the x-line-relaxation step; see §3. For Jacobi and half-Gauss-Seidel line relaxation, the data distribution over R processes in the z-direction allows us to solve systems (n, k_0) and (n, k_1) concurrently if they are mapped to different processes. Systems (n, k) with fixed n mapped to the same process may be solved using a solver that is vectorized in the z-direction.

For Jacobi line relaxation, the data distribution over Q processes in the y-direction allows us to solve systems of a grid plane concurrently. For half-Gauss-Seidel line relaxation, however, the tridiagonal systems within one grid plane $k =$ constant still depend on one another, because system (n, k) depends on $(n-1, k)$. This prevents concurrency within one plane. A further modification allows us to obtain a concurrent program: at the process boundaries in the y-direction, we use old u-values for x-line $(n-1, k)$ in the computation of system (n, k). This *modified half-Gauss–Seidel line relaxation* allows concurrency and vectorization for all grid blocks of the $P \times Q \times R$ process grid.

We shall also consider one alternative to line relaxation. *Segment relaxation* avoids distributed tridiagonal systems by moving terms that couple the system across process boundaries to the right-hand side. For an elementary x-line-relaxation step, this leads to systems with coefficient matrices of the form:

$$T' = \left(\begin{array}{cccccccccc} b_1 & c_1 & & & & & & & & \\ a_2 & b_2 & c_2 & & & & & & & \\ & a_3 & b_3 & & & & & & & \\ \hline & & & b_4 & c_4 & & & & & \\ & & & a_5 & b_5 & c_5 & & & & \\ & & & & a_6 & b_6 & & & & \\ \hline & & & & & & \ddots & & & \\ \hline & & & & & & & b_{M-2} & c_{M-2} & \\ & & & & & & & a_{M-1} & b_{M-1} & c_{M-1} \\ & & & & & & & & b_M & c_M \end{array} \right).$$

Such systems can be solved without any communication and only require solving a tridiagonal system in each process. In §4.2.4, we shall see that, for our application, the price for simplicity is a decreased convergence rate and a lack of robustness.

A discussion of point-relaxation methods is omitted, because they diverge when used for our application. We focus the remainder of this paper on modified half-Gauss-Seidel line, Jacobi line, red-black line, and segment relaxation. Each has a certain convergence rate, which is dependent on the particular application. We postpone a discussion of convergence rates until §4, where our application is introduced.

For simplicity of exposition, the relaxation methods were described with the Poisson equation in mind. Their use in our Navier-Stokes application is more complicated, because every grid point corresponds to four unknowns. In (3), the unknowns u_m and the right-hand side coefficients d_m are vectors of dimension 4. The coefficients a_m, b_m, and c_m are 4×4 matrices. The coefficient matrix T of (4) is block-tridiagonal instead of tridiagonal, and its blocks are 4×4 matrices.

3. Distributed block-tridiagonal solvers. In this section, we shall present several methods to solve sets of block-tridiagonal systems (3) defined on an $M \times N \times K$ computational grid that is distributed over a $P \times Q \times R$ process grid. Each solver can be combined with at least one relaxation method of §2.

In process (p, q, r) of the process grid, we have available an $I \times J \times L$ subgrid of the computational grid. The solution of the discretized Navier-Stokes equations requires alternating-direction line relaxations. Hence, even if the grid is distributed in one dimension only (say $Q = R = 1$ or $P = Q = 1$), process boundaries cross the relaxation lines in at least one direction. We consider only the x-line relaxation with systems of M equations distributed over P processes; the extension to y- and z-line relaxations and to nondistributed line relaxations are obvious.

We shall introduce six different distributed tridiagonal solvers. Each can be converted into a block-tridiagonal solver that can be used to solve the systems that arise in our Navier-Stokes application. To convert, scalars of the tridiagonal solvers must be changed into 4×4 matrices or vectors of dimension 4. Divisions like

$$u_m = \frac{a_m}{b_m}$$

must be replaced by 4×4 systems of equations:

$$b_m u_m = a_m.$$

For the remainder of this section, we treat all coefficients and unknowns as scalars and all line-relaxation systems as tridiagonal. The conversion to block-tridiagonal systems considerably complicates the implementation, but it does not affect the fundamental algorithmic structure.

3.1. Pipelining. The classical sequential direct solver for tridiagonal systems, sometimes referred to as the Thomas algorithm, can be used on distributed data. Although a sequential algorithm, there is sufficient concurrency left, because we must solve many tridiagonal systems. For all practical purposes, this solver can only be used in conjunction with the Jacobi line-relaxation method, for which all tridiagonal systems are independent of one another. In the y- and z-directions, the Jacobi x-line relaxation is concurrent, and the only communication requirement is a boundary exchange.

An elementary x-line-relaxation step uses all processes (p, q, r) with $0 \leq p < P$ and q and r fixed. We now use the J systems of the y-direction to introduce concurrency in the x-direction by a *pipelining* technique (this technique is also studied by Ho and Johnsson [8]).

Assuming that the coefficient matrix T is factored once, the Thomas algorithm solves tridiagonal systems by means of two elementary sequential loops. After data distribution over P processes, these loops remain sequential. In process (p, q, r), we must run through J identical loops. As soon as process (p, q, r) hands over the loop of system j to process $(p + 1, q, r)$, process (p, q, r) may start evaluating the loop of system $j + 1$. Filling the pipeline requires P elementary x-line-relaxation steps.

This procedure is also vectorized in the z-direction: instead of evaluating the loop for one system, all L loops of the z-direction are evaluated. This also reduces communication latency, because messages of L systems are combined into one message. On the other hand, this merging of communication increases the time required for filling the pipeline.

3.2. Concurrent direct tridiagonal solvers. As an alternative to pipelining, it is possible to replace the sequential tridiagonal solver by a concurrent direct tridiagonal solver. This avoids the cost of pipelining. However, all known concurrent direct tridiagonal solvers carry a high floating-point overhead. We have used four different direct solvers. From a fundamental point of view, all four are equivalent. They solve a tridiagonal system by LU-decomposition without pivoting, and their operation counts are very similar. Performance differences are due to technical implementation details.

To identify the different solvers, we follow the nomenclature of Frommer [7] in this paper. Note, however, that there is no generally agreed-upon terminology in the literature. For example, the term "recursive doubling" is used for several methods [7], [15]. We shall use "recursive doubling" and "cyclic reduction" for two variants of the cyclic odd-even reduction algorithm, first proposed by Hockney [9].

3.2.1. Recursive doubling. By eliminating unknowns and equations, one can obtain a new tridiagonal system with half the number of unknowns of the original system. This reduction procedure is repeated until a system is obtained that can be solved immediately.

In the recursive-doubling algorithm, one reduction is computed for each of the M equations. Hence, we can determine all solution components immediately after the last reduction step.

3.2.2. Cyclic reduction. The cyclic-reduction method also performs repeated reductions of the system (3). However, it computes only one solvable equation for the whole system. Subsequently, the solution is computed through back-substitution.

In general, the number of equations per process, I, is greater than one. Therefore, we have to distinguish between a local and a global reduction. In the first $\log_2 I$ steps, we reduce the system such that the unknowns are coupled exclusively with unknowns in neighboring processes. Each tridiagonal system now consists of only one equation per process. In the

$$\left(\begin{array}{ccccccccc}
b_1 & & c_2 & & & & & & \\
& b_2 & c_2 & & & & & & \\
& & b_3 & & & c_3 & & & \\
\hline
& & a_4 & b_4 & & c_4 & & & \\
& & a_5 & & b_5 & c_5 & & & \\
& & a_6 & & & b_6 & & & c_6 \\
\hline
& & & & & a_7 & b_7 & & c_7 \\
& & & & & a_8 & & b_8 & c_8 \\
& & & & & a_9 & & & b_9
\end{array}\right)$$

FIG. 1. *Matrix T after eliminating the upper and lower diagonal entries.*

subsequent global reduction, we eliminate the remaining equations of the first, second, fourth, etc. neighbors.

3.2.3. Partition method. In §3.2.2, the local part of the odd-even reduction reduced the system to a new tridiagonal system with one equation per process. The required communication and the arithmetic overhead of solving the latter system are high. This problem is avoided in Wang's partition method [16].

After an elimination of the upper and lower diagonal entries, the coefficient matrix has the shape given in Fig. 1 for a system with $M = 9$ equations distributed over $P = 3$ processes. The fill that results from this operation can be stored in the original arrays a_m and c_m. This matrix is triangularized and, subsequently, diagonalized. For these operations, Wang transposes T from a row-distributed into a column-distributed matrix. Because this transposition of the matrix requires extensive communication, we follow an alternative procedure. We solve the following tridiagonal system with one equation per process:

$$\begin{pmatrix} b_3 & c_3 & \\ a_6 & b_6 & c_6 \\ & a_9 & b_9 \end{pmatrix} \begin{pmatrix} u_3 \\ u_6 \\ u_9 \end{pmatrix} = \begin{pmatrix} d_3 \\ d_6 \\ d_9 \end{pmatrix}, \tag{5}$$

which is obtained by collecting the equations of the last row in each process. We solved the system (5) by recursive doubling, which we found to be most efficient in the case of one equation per process.

3.2.4. Divide and conquer. In his divide-and-conquer method, Bondeli [2] obtains the solution of (3) by solving a local tridiagonal system in each process concurrently and a global tridiagonal system with one equation in the boundary processes and two equations in the inner processes.

Here, the global problem is solved by eliminating one equation in the inner processes. Again, we obtain a tridiagonal system with one equation per process which is solved with recursive doubling.

3.3. Concurrent iterative tridiagonal solver. The direct solvers of §3.2 solve the system (3) to round-off error (if the systems are well conditioned). Here, we propose a concurrent iterative solver to compute the solution to a certain tolerance.

Splitting the matrix T into a sum $G + H$, as indicated in Fig. 2, the matrix G contains globally uncoupled tridiagonal blocks, and H contains the coefficients that couple the systems across the process boundaries.

This splitting of the coefficient matrix defines the iteration

$$G\,\vec{u}^{(\mu)} = \vec{d} - H\,\vec{u}^{(\mu-1)}. \tag{6}$$

A multicomputer implementation of (6) is easily obtained and is efficient for several reasons:

$$T=\left(\begin{array}{ccc|ccc|ccc}
b_1 & c_1 & & & & & & & \\
a_2 & b_2 & c_2 & & & & & & \\
 & a_3 & b_3 & & & & & & \\ \hline
 & & & b_4 & c_4 & & & & \\
 & & & a_5 & b_5 & c_5 & & & \\
 & & & & a_6 & b_6 & & & \\ \hline
 & & & & & & b_7 & c_7 & \\
 & & & & & & a_8 & b_8 & c_8 \\
 & & & & & & & a_9 & b_9
\end{array}\right)+\left(\begin{array}{ccc|ccc|ccc}
 & & & & & & & & \\
 & & & & & & & & \\
 & & & c_3 & & & & & \\ \hline
 & & a_4 & & & & & & \\
 & & & & & & & & \\
 & & & & & & c_6 & & \\ \hline
 & & & & & a_7 & & & \\
 & & & & & & & & \\
 & & & & & & & &
\end{array}\right)$$

FIG. 2. *Splitting of T for $M = 9$ and $P = 3$.*

1. Because G contains only the uncoupled tridiagonal matrices, its inversion is trivially concurrent. The LU-decomposition of G needs to be carried out only once. We can solve (6) by computing the right-hand side terms and by evaluating two back-substitution loops in each process.

2. The matrix-vector operation $\vec{d} - H\,\vec{u}^{\,(\mu-1)}$ requires only nearest-neighbor communication.

3. Only the right-hand side terms of the first and the last row in each process change within one iteration step. Hence, the arithmetic costs of one iteration step are low.

4. A good initial guess for the iteration is found by solving the system

$$G\,\vec{u}^{\,(0)} = \vec{d}. \tag{7}$$

5. The iteration (6) is continued until some stopping criterion is satisfied. This criterion can be chosen to fit the need and, usually, depends on the specific application; see §4.2.5 for a further discussion.

4. Computational results. In this section, we compare the performance on the Intel Touchstone Delta of the relaxation methods of §2 implemented using the block-tridiagonal solvers of §3. All performance data are obtained for a solver of the Navier–Stokes equations for three-dimensional, incompressible, unsteady, and viscous flows. In §4.1, we describe the problem in sufficient detail to understand the computational complexity of the application. However, fluid-dynamical results obtained with this code are published elsewhere [1]. Additional physical and numerical details are found in [3] and [4].

Our problem sizes and convergence properties are not artificial creations. The problem size was determined by realistic resolution requirements and was NOT increased to obtain artificially high efficiency typically associated with coarse-grain computations. Similarly, our selection of numerical methods, i.e., of line-relaxation methods over point-relaxation methods, was guided by realistic needs. There is no compelling reason to use line-relaxation methods for the Poisson equation, because one can always use simple point-relaxation methods. This alternative is not available now, because point-relaxation methods do not even converge for our application. In §4.2.4, we shall also observe that segment relaxation is not sufficiently robust for our Navier–Stokes problem. It is very important that our performance measurements are obtained for a "real" problem. The grid size and the accuracy with which the discrete problems are solved are dictated by the physics of our application.

In our performance analysis of concurrent relaxation methods, we distinguish between numerical and implementation issues. Numerical issues primarily determine the number of relaxation-iteration steps necessary for convergence. This is, of course, extremely application dependent. Implementation issues primarily determine the execution time of one relaxation step through cost of communication, load balance, number of processes, etc. This depends on the application through computational parameters like grid dimensions and number of unknowns. Nevertheless, the performance results of one relaxation step are more readily transferred to other applications. For this reason, we consider these two performance issues separately.

We shall present the convergence rates for the different relaxation methods in §4.2. Multicomputer performance of one alternating-direction line-relaxation step will be discussed in §4.3. Convergence-rate and per-step-performance information are combined into global-performance results in §4.4.

4.1. Discretization of the Navier–Stokes equations. The Navier–Stokes equations are a set of coupled nonlinear partial-differential equations, which describe the conservation of mass, momentum, and energy for continuous media. For an incompressible fluid, a time-dependent flow in three dimensions is defined by the pressure $p(x, y, z, t)$ and the three velocity components $u(x, y, z, t)$, $v(x, y, z, t)$, and $w(x, y, z, t)$. In a dimensionless matrix-vector form, the conservation equations are given by

$$\bar{R}\cdot\frac{\partial Q}{\partial t}+\frac{\partial F}{\partial x}+\frac{\partial G}{\partial y}+\frac{\partial H}{\partial z}=\frac{1}{Re}\cdot\left(\frac{\partial^2 S}{\partial x^2}+\frac{\partial^2 S}{\partial y^2}+\frac{\partial^2 S}{\partial z^2}\right), \tag{8}$$

where

$$\bar{R}=\begin{pmatrix}0&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix},\quad Q=\begin{pmatrix}p\\u\\v\\w\end{pmatrix},\quad S=\begin{pmatrix}0\\u\\v\\w\end{pmatrix},$$

$$F=\begin{pmatrix}u\\u^2+p\\uv\\uw\end{pmatrix},\quad G=\begin{pmatrix}v\\vu\\v^2+p\\vw\end{pmatrix},\quad H=\begin{pmatrix}w\\wu\\wv\\w^2+p\end{pmatrix}.$$

Breuer and Hänel [4] extended the method of artificial compressibility to unsteady flows. They define an artificial time τ and add a supplementary time derivative $\hat{R}\frac{\partial Q}{\partial \tau}$, with

$$\hat{R}=\begin{pmatrix}\frac{1}{\beta^2}&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix},$$

to the Navier–Stokes equations. The pressure field is now coupled to the velocity distribution, and (8) can be integrated. Because a steady solution is computed at time τ, the additional terms vanish, and we obtain the unsteady solution of (8) at the physical time t.

Let ξ be an index for the physical time step and ζ an index for the artificial time step. The approximation

$$\frac{\partial Q}{\partial \tau}\approx\frac{Q^{\xi+1,\zeta+1}-Q^{\xi+1,\zeta}}{\Delta\tau}=\frac{\Delta Q^{\zeta}}{\Delta\tau}$$

leads to an implicit discretization scheme for the artificial time step. In this scheme, the Euler and viscous fluxes are defined at the new physical time step $\xi+1$ and at the new artificial time step $\zeta+1$. Consider, e.g., the Euler flux F. The implicit discretization uses its value $F^{\xi+1,\zeta+1}$ at physical time step $\xi+1$ and artificial time step $\zeta+1$. This term is expanded in a Taylor series with respect to the artificial time τ to obtain the following linearization:

$$F^{\xi+1,\zeta+1}=F^{\xi+1,\zeta}+J_F^{\xi+1,\zeta}\Delta Q^{\zeta}.$$

Here, J_F is the Jacobian matrix of the Euler flux F with respect to Q, say $J_F = \frac{\partial F}{\partial Q}$. The resulting discrete system of equations is given by

$$\text{LHS} \cdot \Delta Q^{\zeta} = \text{RHS}\,. \tag{9}$$

The right-hand side contains the expressions that arise from the discretization of the derivatives in (8) using a high-order upwind scheme for the convective terms, central differences for the diffusive terms, and a second-order discretization for the physical time t. The left-hand side contains first-order spatial derivatives of the Jacobian matrices.

Because the left-hand side terms in (9) vanish for the exact solution vector Q, the discrete solution has the order of accuracy of the right-hand side. The numerical advantage is that the spatial discretization of the Jacobian matrices has no influence on the order of accuracy of the solution. Therefore, we have gained some flexibility in choosing a discretization of LHS. This flexibility is used to improve the diagonal dominance of the coefficient matrix and to increase the size of the time step.

Although central differences would be the most straightforward choice, they do not add terms to the main diagonal of the coefficient matrix. That is why we use a first-order upwind formulation combined with a matrix-splitting technique: after splitting the Jacobian matrices into positive and negative parts, forward differences are applied to the positive parts and backward differences to the negative parts. Consider, e.g., the Jacobian matrix J_F. This 4×4 matrix is diagonalized such that

$$J_F = M \Lambda M^{-1},$$

where M is the matrix of eigenvectors of J_F, and Λ is the diagonal matrix containing the eigenvalues of J_F on its diagonal. Let

$$\Lambda = \Lambda^{+} + \Lambda^{-},$$

where Λ^{+} and Λ^{-} contain, respectively, the positive and negative eigenvalues. After transforming back to the Jacobian J_F, we obtain the splitting

$$J_F = M\Lambda^{+}M^{-1} + M\Lambda^{-}M^{-1} = J_F^{+} + J_F^{-}.$$

The spatial derivatives in J_F^{+} and J_F^{-} are discretized, respectively, with forward and backward differences. This method tends to increase the diagonal entries of the coefficient matrix, and the resulting system of equations can be solved with a line-relaxation method. For further details and for a validation of the above scheme, refer to Breuer [3] and Breuer and Hänel [4].

Every solution component consists of four elements: the pressure and the three velocity components. Hence, when applying the methods described in §3, the system (3) becomes a block-tridiagonal system, the coefficients a_m, b_m, and c_m are 4×4 matrices, and the right-hand side terms d_m are vectors consisting of four components. It follows that the discretized Navier-Stokes equations require much more arithmetic per grid point than, e.g., the discretized Poisson equation.

All computations of this paper use the above method to simulate the flow of an isolated slender vortex embedded in an axial flow. The scientific interest in this flow is the study of the phenomenon of a bursting vortex or vortex breakdown. Although the initial conditions are axially symmetric, the flow is fully three-dimensional after vortex breakdown. The boundary conditions in the outflow plane and on the lateral boundaries are extrapolated from the case of a free vortex. At the outflow plane, a nonreflecting boundary condition simulates undisturbed vortical flow through the boundary. At the lateral boundaries, the pressure is imposed such

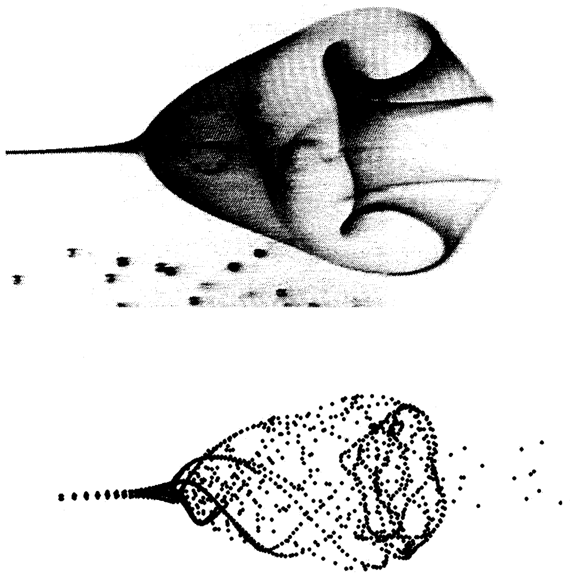


FIG. 3. *Experimental and numerical flow visualization of bubble-type vortex breakdown: top picture shows experimental streaklines and bottom picture shows numerical streaklines.*

that the bursted part of the vortex lies well within the domain of integration. Breuer [3] gives complete details on the initial and on the boundary conditions required to simulate this flow.

Figure 3 compares a numerical simulation and an experiment of Althaus et al. [1]. Using a Reynolds number based on the radius of the vortex core as length scale and the mean axial velocity as velocity scale, both cases have a Reynolds number of 500. Our simulation uses a $32 \times 32 \times 64$ rectangular grid with the z-direction being the axial coordinate. The physical time step is 0.3, and the artificial time step is 100. Althaus et al. [1] perform a detailed comparison of numerical and experimental results. It is this computation that we use to evaluate the performance of all our codes.

Depending on the orientation of the relaxation line, we have to solve $32 \times 64 = 2048$ systems of 32 blocks or $32 \times 32 = 1024$ systems of 64 blocks of size 4×4. Our challenge is to make effective use of all 512 computing nodes of the Delta to solve the block-tridiagonal systems of this moderate size. As discussed in §2, we use a $P \times Q \times R$ process grid. Preliminary tests, which we do not report, showed that three-dimensional grid distributions outperform one- and two-dimensional grid distributions with $P = 1$ and/or $Q = 1$, in spite of the fact that the latter can use efficient sequential tridiagonal solvers in at least one direction. This may seem somewhat counterintuitive. Given a certain number of nodes, choosing $P = 1$ leads to higher values for Q and/or R. The fact that x-line-relaxation steps are very efficient when $P = 1$ is offset by the increased load imbalance and the decreased efficiency in the other directions. Therefore, we always use a process grid with $P \approx Q \approx R$; see Table 1. In all our computations, each process is mapped to a separate node of the multicomputer.

The experiments of this section were performed on the Intel Touchstone Delta [5], which consists of an ensemble of 512 computing nodes connected in a two-dimensional mesh. The computational engine of each node is an Intel i860 processor with an advertised peak perfor-

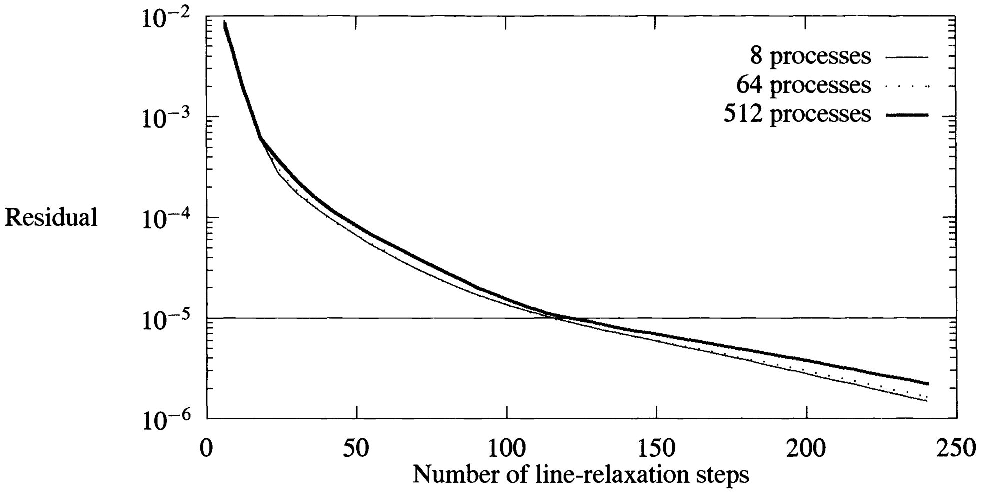


FIG. 4. *Maximal residual as a function of the number of modified half-Gauss–Seidel line-relaxation steps for three different process grids.*

TABLE 1
Process grids used in our computational tests.

Total	Number of processes in		
Number	x-direction (P)	y-direction (Q)	z-direction (R)
4	1	1	4
8	2	2	2
16	2	2	4
32	2	2	8
64	4	4	4
128	4	4	8
256	4	4	16
512	8	8	8

mance of 60 double-precision MFLOPS. Each node has a local memory of 16 MB. On the Delta, our standard problem needs a minimum of four nodes because of memory requirements.

4.2. Convergence of relaxation methods.

4.2.1. Modified half-Gauss–Seidel line relaxation. The modified half-Gauss-Seidel line relaxation uses the updated unknowns of the previously computed plane except at process boundaries; see §2. The convergence rate of the modified method is dependent on the number of processes and on the choice of process boundaries, because they determine the matrix splitting underlying the relaxation method. Figure 4 shows the residual as a function of the number of relaxations for some of the partitions of Table 1. Although the convergence depends on the partition, all computations achieve the required residual of 10^{-5} within approximately the same number of iteration steps.

4.2.2. Jacobi line relaxation. In the case of Jacobi line relaxation, all relaxation lines may be computed simultaneously. The convergence rate of the Jacobi line relaxation does not depend on the number of processes. Note that the modified half-Gauss-Seidel line relaxation turns into the Jacobi line relaxation when the number of processes is equal to the number of grid planes.

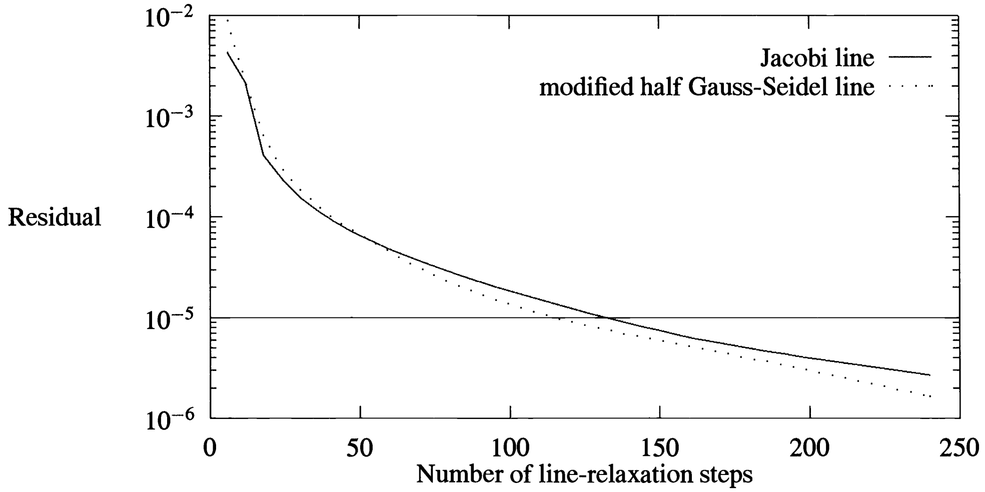


FIG. 5. *Maximal residual as a function of the number of line-relaxation steps in a computation with* 64 *processes.*

In Fig. 5, half-Gauss-Seidel and Jacobi line relaxation on the 64-node partition are compared. The Jacobi method needs about 10% more iteration steps to reach the required tolerance on the residual.

4.2.3. Red-Black line relaxation. Like the Jacobi line relaxation, the convergence rate of the Gauss-Seidel line relaxation with red-black ordering does not depend on the number of processes. We obtained a test program for the red-black line relaxation easily by splitting the Jacobi relaxation into the two steps explained in §2 and changing the step width of the inner loops to two. After the first step, the updated unknowns have to be exchanged between the processes. Besides the required additional communication, there is a loss of performance, because the inner loops can no longer be vectorized.

For our application, there is no observable difference in the convergence rate between the red-black and the Jacobi line iteration. The red-black line relaxation is, therefore, not a viable alternative to the modified half-Gauss-Seidel line method and, because of the communication and vectorization losses, its global performance is worse than the Jacobi line relaxation.

4.2.4. Segment relaxation. The segment relaxation computes relaxation lines interrupted by the process boundaries. Efficiency and applicability of this method depend on the convergence rate of the resulting iteration. Consider a segment-relaxation step in one direction with M grid points distributed over P processes, and let

$$I = \frac{M}{P}$$

be the number of local grid points. For $I = M$, we obtain a sequential line relaxation, while a Jacobi point relaxation results if $I = 1$.

We implemented an alternating-direction segment relaxation and plot the residual as a function of the number of relaxation steps for a 256- and a 512-node partition in Fig. 6. The plot shows that the relaxation fails to converge for the unbalanced $4 \times 4 \times 16$ partition. The $8 \times 8 \times 8$ partition needs about 20% more iteration steps than the modified half-Gauss-Seidel line relaxation.

We observed a nonconverging iteration even for the $2 \times 2 \times 8$ partition. We conclude that segment relaxation is not suitable, because robustness of the algorithm cannot be guaranteed.

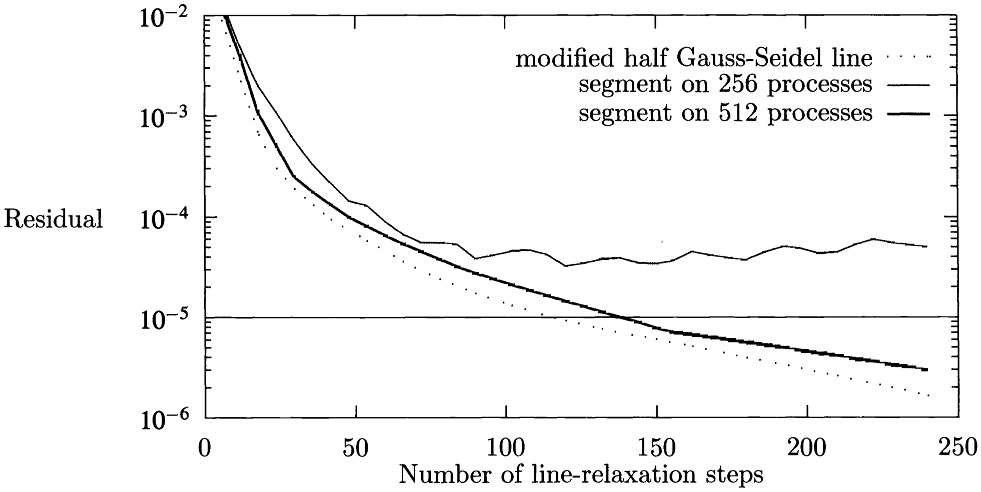


FIG. 6. *Maximal residual as a function of the number of line-relaxation steps for segment relaxation and modified half-Gauss–Seidel line relaxation.*

Although we achieve convergence when using all nodes on the Delta, it is uncertain whether the code would converge on a larger configuration of the same hardware. We consider this nonrobust behavior unacceptable.

4.2.5. Iterative block-tridiagonal solver. The modified half-Gauss–Seidel line relaxation of §4.2.1 can also be implemented using the concurrent iterative block-tridiagonal solver of §3.3. Because this method solves the system (3) only approximately, one should consider the effect of less-accurate block-tridiagonal solvers on the convergence rate of the line-relaxation method.

In Fig. 7, we examine the impact on the number of line-relaxation steps of using the iterative solver with six block-tridiagonal iteration steps. (The rationale for using six iteration steps is discussed in §4.3.6.) The curves for the exact and iterative solvers show the same progress, and the required residual of 10^{-5} is reached almost within the same relaxation step.

With an increasing number of processes, the initial guess (7) gets worse, and more coupling coefficients are set equal to zero to obtain the tridiagonal-iteration matrix. The partition, therefore, has an impact on the convergence rate. Figure 8 compares three different partitions. Up to 256 processes, the curves virtually coincide. The computation with 512 nodes requires only a few additional line-relaxation steps to converge.

4.3. Performance of one alternating-direction line-relaxation step. In this section, we shall compare the multicomputer performance of one alternating-direction line-relaxation step, i.e., one line relaxation in each spatial direction. To obtain a dimensionless efficiency for a multicomputer program, one must relate the execution time to a meaningful reference time. In principle, this reference time must be the best execution time possible on one node of the same multicomputer. In reality, one must settle for the sequential-execution time of a reasonable but not necessarily the best sequential program. For us, even the reasonable sequential time is impossible, because our problem requires too much memory to solve on one node. We defined a fictitious reference time in the following manner. To obtain a sequential time for the relaxation in the x-direction, we distribute the grid only in the y- and z-directions over the processes. Subsequently, the concurrent-execution time for this partition is then multiplied by the number of processes. We repeat this method for the y- and z-directions. Our

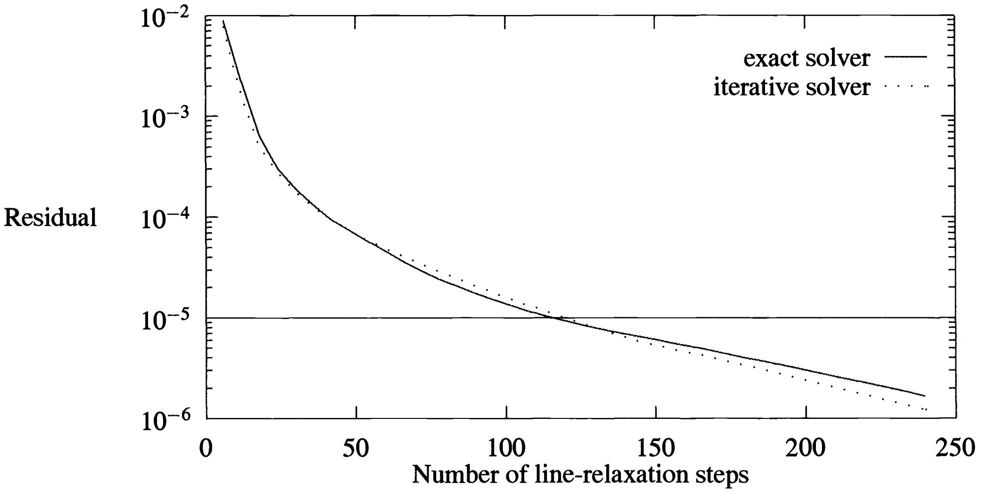


FIG. 7. *Maximal residual as a function of the number of line-relaxation steps when using the exact and the iterative block-tridiagonal solver to implement modified half-Gauss–Seidel line relaxation with* 64 *processes.*

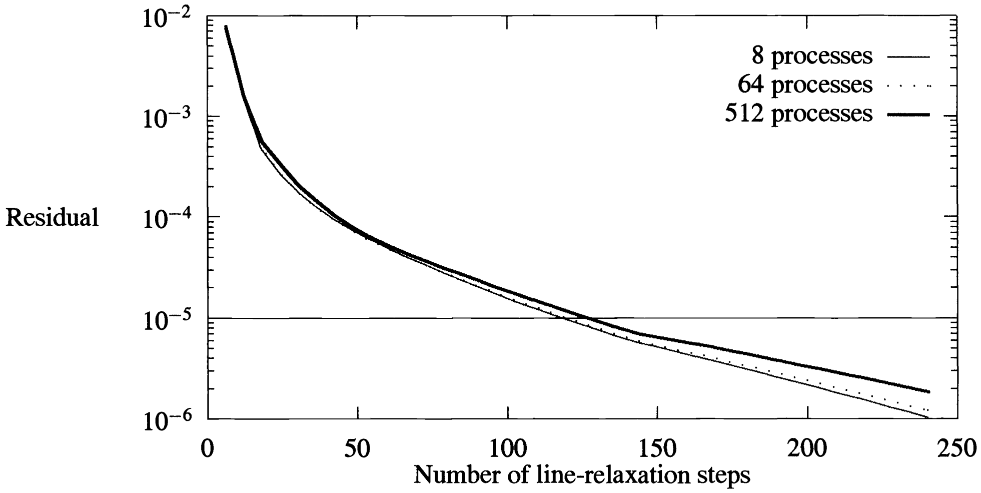


FIG. 8. *Maximal residual as a function of the number of line-relaxation steps to examine impact of process grids on the iterative block-tridiagonal solver.*

sequential-execution time for the alternating-direction line relaxation is the sum of these three terms. With process boundaries that are parallel to the relaxation lines, the block-tridiagonal systems can be solved by sequential LU-decomposition, which is the optimal procedure for sequential computations.

The same reference time is used for all methods, such that the efficiency results can be compared directly. The efficiency η_P of a P-node computation is the ratio

$$\eta_P = \frac{\text{(reference sequential time)}}{P \cdot \text{(execution time of the } P\text{-node program)}}.$$

We shall compare the multicomputer performance by means of two different kinds of plots. The first kind plots the efficiency η_P as a function of the number of nodes, and the second plots the execution time T_P in seconds as a function of the number of nodes. In both plots, a logarithmic scale is used for the number of nodes. In the execution-time plot, we also use a logarithmic scale for the second axis such that the line of linear speedup has slope -1. (The slope is distorted, however, because different scaling factors are used for the horizontal and vertical axes.)

The increased computational requirements of the discrete Navier-Stokes equations over, e.g., the discrete Poisson equation actually results in a more efficient computation, because more arithmetic occurs for the same number of messages. Although the messages are longer for the discrete Navier-Stokes equations, the latency usually dominates the communication time, and the length of the messages is less important than the number of messages.

4.3.1. Pipelining. In §3.1, we described a pipelining method for the sequential LU-decomposition. This makes sense if the number of processes is small compared with the number of unknowns per block-tridiagonal system [8]. This method, compared with all other methods we implemented, has the lowest floating-point operation count and requires the least amount of communication. However, it is less concurrent than some other methods because of load imbalance. The pipelining implements the Jacobi line relaxation.

Figure 9 shows the performance of the pipelining algorithm. As expected, the algorithm shows good efficiency for a small number of processes. However, for larger numbers of processes, the startup time required for all processes to participate in the computation causes a progressive loss of efficiency.

4.3.2. Recursive doubling. A concurrent implementation of the modified half-Gauss-Seidel line relaxation with the recursive-doubling algorithm for block-tridiagonal systems requires more arithmetic and more communication than the pipelining method, but offers higher concurrency. The recursive-doubling computations in the execution-time plot of Fig. 10 lie on a line that is almost parallel to the line of linear speedup. These lines are parallel and the efficiency is constant as a function of the number of nodes, because the communication overhead plays only a secondary role in computations with up to 512 nodes. The vertical distance between these two lines can be related to the arithmetic overhead, which is mainly responsible for the disappointing efficiency of about 15%.

4.3.3. Cyclic reduction. The cyclic-reduction algorithm is based on the same reduction procedure as recursive doubling and is also an implementation of the half-Gauss-Seidel line relaxation. This method needs less arithmetic, but additional communication for the back-substitution. Whereas the local part of the cyclic reduction shows a good load balance, the number of processes that participate during the global part halves after each reduction step. Both facts lead to a significant loss of efficiency for large numbers of processes (Fig. 10). However, cyclic reduction always beats recursive doubling. It beats the pipelining method for computations with all available nodes (see also Table 2).

For the back-substitution, the cyclic-reduction algorithm must store the entries of the coefficient matrix T and the right-hand side terms in (3), which change after each reduction step. The additional memory made a run on four nodes impossible.

The dips in the efficiency curves for 32 and 256 processes occur, because we double the partitions in the x- and y-directions (see Table 1). In these cases, the z-line-relaxation steps are inefficient compared to the next larger partition ($2 \times 2 \times 8 \rightarrow 4 \times 4 \times 4$ and $4 \times 4 \times 16 \rightarrow 8 \times 8 \times 8$, respectively).

4.3.4. Partition method. When modified half-Gauss-Seidel line relaxation is implemented by means of Wang's partition method, we realize an improved efficiency compared with the cyclic-reduction algorithm; compare Figs. 10 and 11. We examined in detail execution profiles of the z-line relaxation with a $1 \times 1 \times 8$ process grid. In this case, already

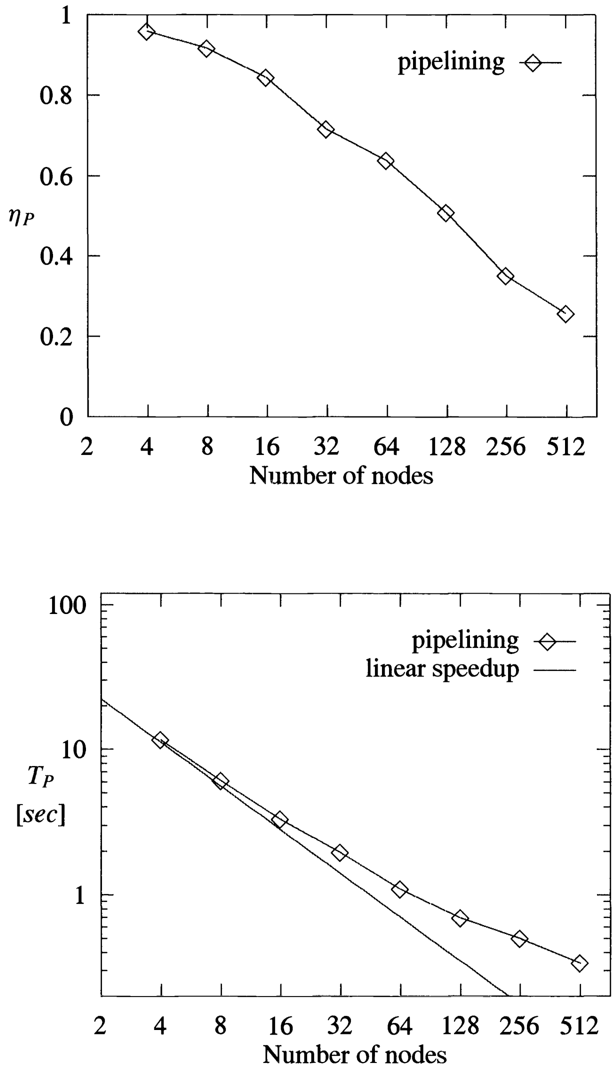
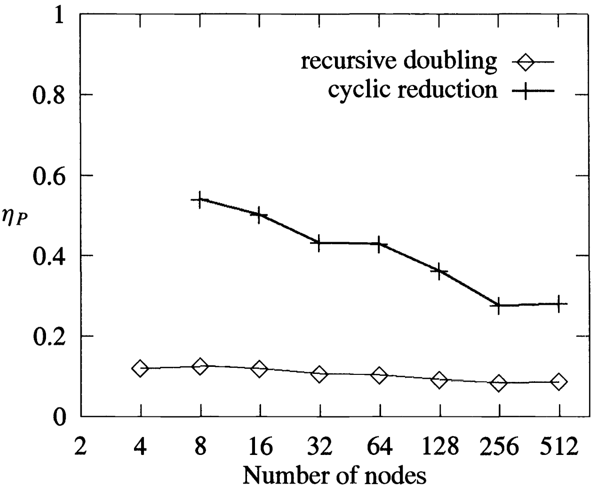


FIG. 9. *Efficiency and execution time of the pipelining algorithm.*

about 40% of the execution time was needed for solving the reduced system computed in the first part of Wang's partition method. This step is communication intensive and becomes even more so when the number of processes is increased further. It is this part of the computation that is responsible for the gradual loss of efficiency as the number of processes is increased.

4.3.5. Divide and conquer. The performance of the divide-and-conquer algorithm applied to the modified half-Gauss-Seidel line-relaxation method is given in Fig. 11. Like the partition method, divide and conquer reduces the size of the tridiagonal systems down to one block of equations per process.

The communication needed for the solution of the reduced system, obtained again with recursive doubling, decreases the efficiency of larger partitions in our application. The performance as a function of the number of processes is almost identical to that of the partition



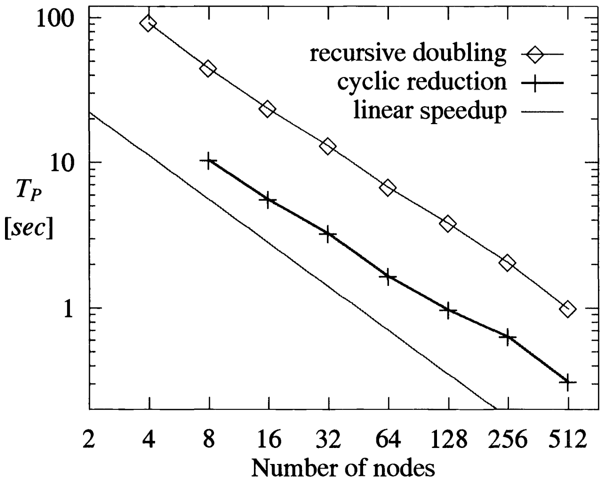


FIG. 10. *Efficiency and execution time of recursive doubling and cyclic reduction.*

TABLE 2
Execution times until convergence on 512 *nodes.*

Block-tridiagonal solver	Execution time in seconds
Iteration method	10.12
Partition method	11.08
Divide and conquer	12.31
Cyclic reduction	12.44
Pipelining	14.96
Recursive doubling	39.64

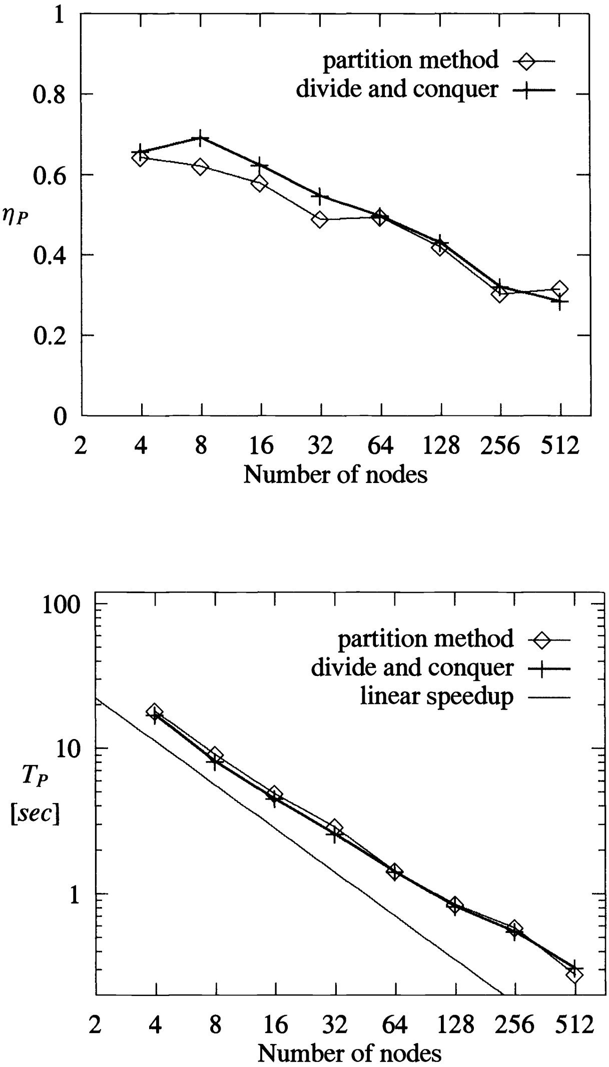


FIG. 11. *Efficiency and execution time of the partition and the divide-and-conquer method.*

method. The divide-and-conquer algorithm outperforms the partitioning method on smaller process grids.

4.3.6. Iterative block-tridiagonal solver. With the iterative block-tridiagonal solver, we implemented the modified half-Gauss-Seidel line relaxation. Because the iteration method solves the block-tridiagonal systems approximately, the accuracy requirements are a decisive determinant for the performance of the iteration method. In principle, it would be possible to iterate until a certain criterion is satisfied, e.g.,

$$|\vec{u}^{(\mu)} - \vec{u}^{(\mu-1)}| \leq \varepsilon, \tag{10}$$

where ε is a certain tolerance. If ε is small, the iterative solver is numerically equivalent to a direct solver. However, if ε is large, the error on the solution of the tridiagonal system may have an impact on the convergence rate of the line relaxation. The size of ε in (10), therefore, not only determines the number of block-tridiagonal iteration steps, but also the number of line-relaxation steps.

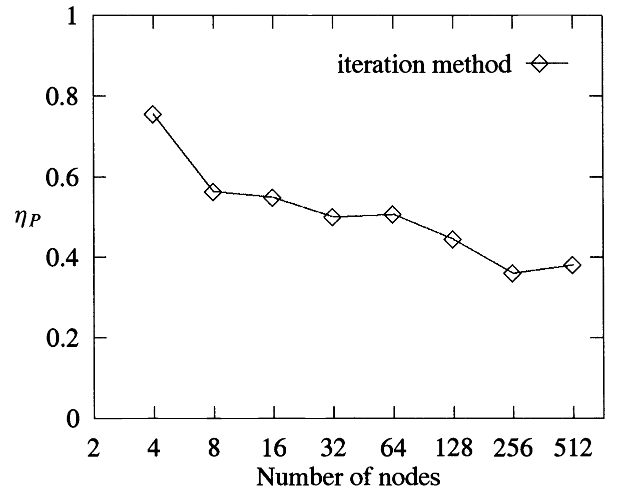
In a multicomputer computation, error-adaptive strategies like those suggested by (10) add significant costs that are difficult to recoup by the expected reduction in the number of iteration steps: computing error estimates is expensive, because they require global communication. Therefore, (10) is not used in our final computation. Instead, we are using a fixed number of block-tridiagonal iteration steps. We found that this is more efficient, while highly reliable if the number of block-tridiagonal iteration steps is sufficiently large. To determine an appropriate number, we performed a sequence of experiments on a $1 \times 1 \times 8$ process grid. In (10), we set $\varepsilon = 10^{-6}$, which is one order of magnitude below the requested residual of the global relaxation. We found that six iterations were always sufficient to reach the required accuracy if the initial guess $\vec{u}^{(0)}$ of (7) was used. The performance results shown in Fig. 12 were all measured with the number of block-tridiagonal iteration steps set equal to 6. With more than 32 processes, the algorithm is more efficient than the partition method and more efficient than the divide-and-conquer technique. With 256 processes, the performance is better than any other concurrent algorithm we implemented.

It was already shown in §4.2.5 that, with six block-tridiagonal iteration steps, there was virtually no difference in the number of line-relaxation steps between the iterative and the exact block-tridiagonal solver. Line-relaxation steps based on either solver are, therefore, practically equivalent. It is possible to reduce the number of block-tridiagonal iteration steps below six. However, the resulting line-relaxation method is no longer equivalent with the original method and may require a larger number of line-relaxation steps. On the other hand, each line-relaxation step may be considerably less expensive. We did not pursue this possibility of trading off the accuracy of the block-tridiagonal solver with the convergence rate of the line-relaxation method. However, segment relaxation is equivalent to line relaxation with one iteration step of the iterative block-tridiagonal solver. Considering the robustness problems of segment relaxation, at least two block-tridiagonal iteration steps are needed.

4.4. Global performance. The interesting performance of a program is, of course, the execution time until convergence within tolerance. In Fig. 13, we compare this time for the Jacobi line-relaxation method implemented with pipelining and for the modified half-Gauss–Seidel line relaxation implemented with the partition method and with the iterative block-tridiagonal solver. Here, we focus only on the execution time needed for the relaxation routines. To obtain the actual time to solve the Navier–Stokes equations, we must add to this the time for the computation of the right-hand side terms and the boundary conditions, which are independent of the solution procedure.

The better performance of the iterative block-tridiagonal solver for computations with more than 256 processes is partially offset by the required additional relaxation-iteration steps; see §4.2.5. The convergence losses of the Jacobi line relaxation (§4.2.2) deteriorate the performance of the pipelining algorithm. The execution times until convergence on 512 nodes for all implementations are given in Table 2. The modified half-Gauss–Seidel line relaxation implemented with the iterative block-tridiagonal solver clearly beats all competitors. As pointed out before, there are several techniques that could improve its performance even further. Most importantly, the fastest method is also, by far, the easiest to implement.

5. Computer dependence. The execution time of a program is, by its very nature, a computer-dependent characteristic. One must, therefore, always question whether or not particular performance results for a set of algorithms obtained on one computer carry over to other computers. Although not our main concern here, comparative performance studies can also be used for computer-evaluation purposes. In a preliminary comparative study, we obtained



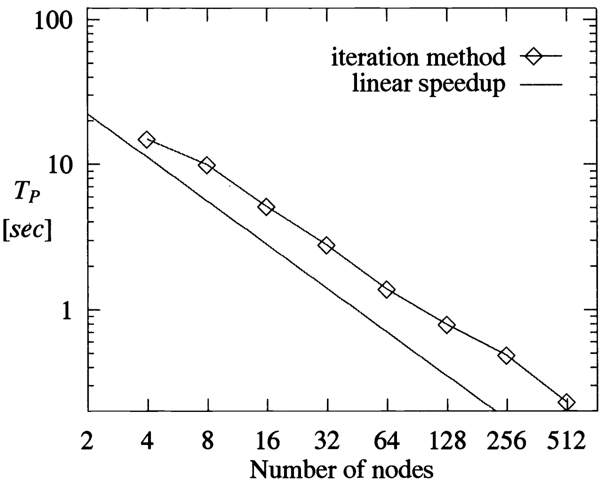


FIG. 12. *Efficiency and execution time of the iteration method.*

some performance data on the Intel Paragon XP/S [6] and the Parsytec SuperCluster [13]. We also used the Fujitsu S-600, which is a conventional vector processor with an advertised peak performance of five GFLOPS.

The Paragon design is similar to that of the Delta. The i860 processors of the Paragon have a higher clock frequency than those of the Delta: 50 MHz instead of 40 MHz. In principle, this increases the advertised peak performance by 25% to 75 MFLOPS per node. Furthermore, Paragon nodes have larger caches and contain improved communication hardware to reduce latency and increase bandwidth.

The Parsytec SuperCluster consists of 256 Transputers T-805, which are clocked at 30 MHz. Each Transputer has an advertised peak performance of 2.2 MFlops. With 4 MBytes of memory per node, our application requires at least 16 Parsytec nodes. Each node contains four bidirectional communication links, which can be used to configure the nodes into a large

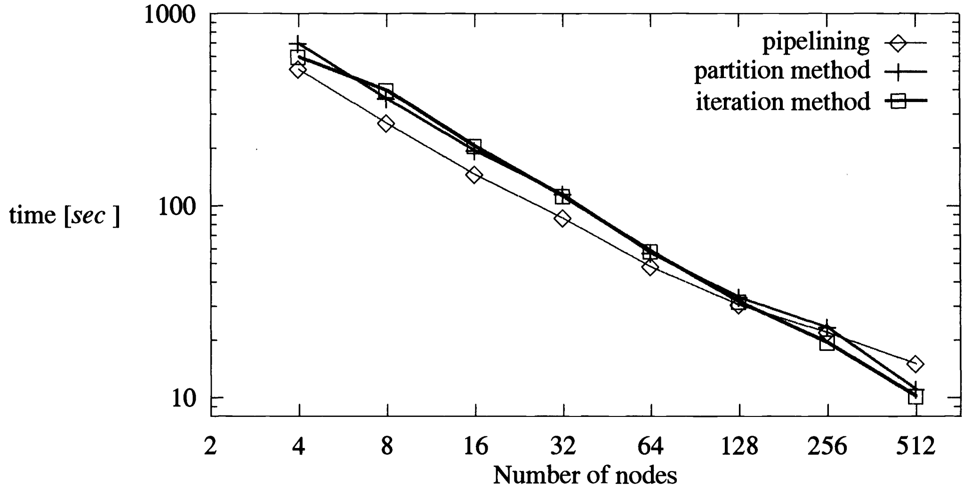


FIG. 13. *Execution time until residual* $\leq 10^{-5}$.

variety of network topologies. The floating-point performance of the Transputer is considerably less than that of a Paragon or a Delta node. However, the Parsytec has an excellent ratio of communication versus arithmetic time.

In Figs. 14, 15, and 16, the execution time of one alternating-direction line-relaxation step is displayed as a function of the number of nodes. Figure 14 considers Jacobi line relaxation implemented using the pipelined block-tridiagonal solver on the Delta, Paragon, and Parsytec. Figures 15 and 16 display execution times obtained with modified half-Gauss-Seidel line relaxation. Figure 15 is for the program based on the concurrent iterative block-tridiagonal solver, and Figure 16 for the program that uses the partition method to solve the block-tridiagonal systems concurrently. In these three figures, the lines of linear speedup are based on a sequential-execution time obtained for the Delta as explained in §4.3. The lines of linear speedup for Paragon and Parsytec are parallel to this line, but are not displayed to avoid crowding the figures; Parsytec efficiency is compared with Delta efficiency in Fig. 17.

In the execution-time plots, the Parsytec computations lie on a line that is almost parallel to the line of linear speedup. This is most pronounced for the method based on the iterative block-tridiagonal solver. This is an indication that almost all overhead on the Parsytec is due to the increased operation count of the concurrent computations and not to communication. This is confirmed in the efficiency plots of Fig. 17, which displays the efficiencies as a function of the number of nodes. (Efficiencies are computed using the sequential-execution time on the computer of the concurrent computation.) The iteration method on the Parsytec has a constant efficiency of about 40%, which indicates that communication overhead is negligible. When executed on one node, the concurrent Parsytec program is about $\frac{1}{0.4} = 2.5$ times slower than the sequential Parsytec program. However, the concurrent program speeds up linearly.

The pipelining method does not have a constant efficiency on the Parsytec. However, its decrease in efficiency is much less pronounced than on the Delta: communication and load-imbalance effects are not as important on the Parsytec as on the Delta. This is, of course, easily explained by the smaller communication-arithmetic ratio of the Parsytec.

Because of the low floating-point-operation count of the pipelining method and the high communication efficiency of the Parsytec, pipelining remains the fastest algorithm for up to the maximum available number of nodes (256). However, extrapolating Fig. 17, we expect that

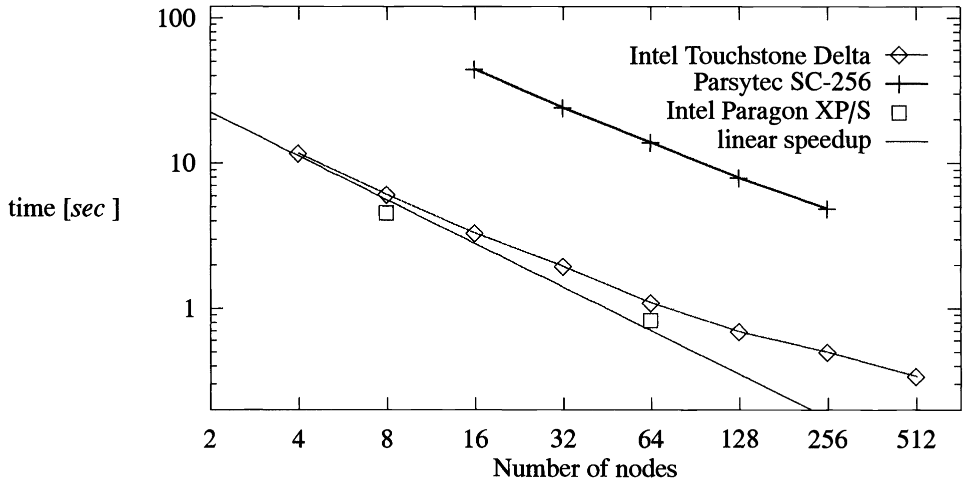


FIG. 14. *Execution time of the pipelining method.*

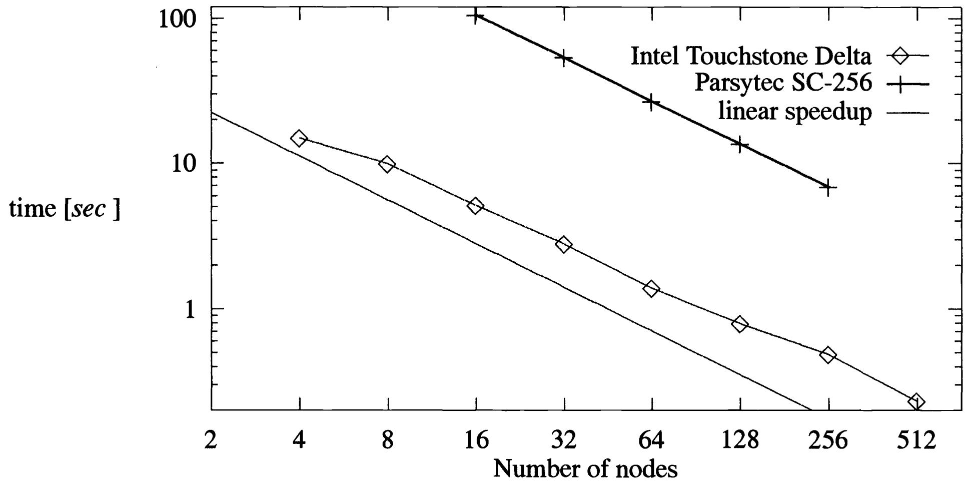


FIG. 15. *Execution time of the iteration method.*

the iteration method will perform better than the pipelining method on Parsytec-like computers with more than 512 nodes.

For our computations, the Delta is about 10–13 times faster than the Parsytec. Based on the advertised peak performance of their nodes, the Delta should be about 30 times faster than the Parsytec. The speed difference between Parsytec and Delta is smaller for fine-grain computations, because the Parsytec is a more efficient computer. The higher efficiency is not enough, however, to close the speed gap with the Delta.

As mentioned before, our code solves the Navier–Stokes equations to simulate time-dependent, incompressible, and unsteady flows. Table 3 displays typical execution times to obtain useful fluid-dynamical results. These require about 1000 physical time steps, and each time step requires many line-relaxation steps. These execution times also include all other computations that surround the actual alternating-direction line-relaxation iteration. The most

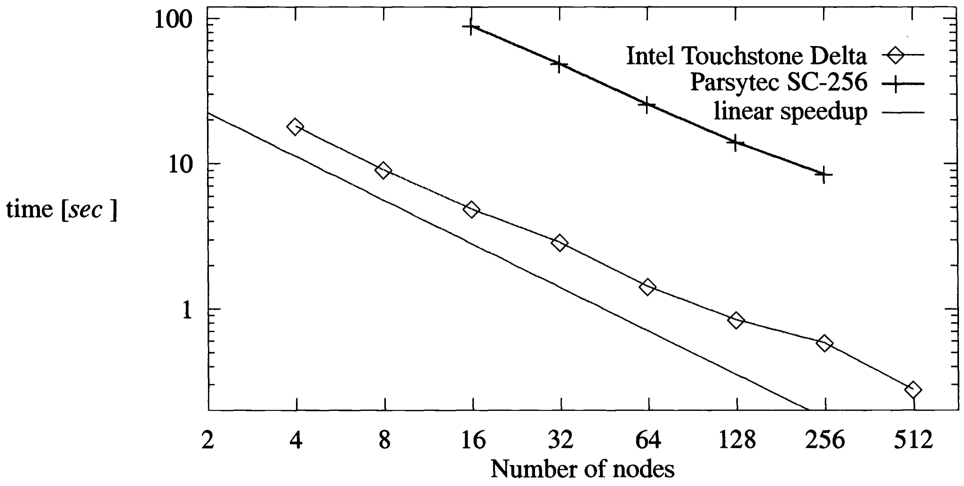


FIG. 16. *Execution time of the partition method.*

TABLE 3
Typical execution times for the production of fluid-dynamical results.

Computer	Execution time in hours on			
	8 nodes	64 nodes	256 nodes	512 nodes
Intel Paragon XP/S	70.8	14.3	-	-
Intel Touchstone Delta	97.2	18.2	9.1	8.4
Parsytec SC-256	-	235.2	84.2	-
Fujitsu S-600	13.1			

important factor in these periferal-but-necessary computations is the evaluation of the boundary conditions. For our application, the Fujitsu S-600 is equivalent to about 64 Paragon nodes. Whereas an individual line-relaxation step is almost four times faster on 256 Delta nodes than on 64 Delta nodes, only a factor of two is obtained in the global computation. This is due to the increasing influence of the evaluation of the boundary conditions, which introduces significant load imbalance in finer-grain computations.

6. Summary. We presented several concurrent methods for solving sets of block-tridiagonal systems on a rectangular grid. Previous studies concentrated on computations of very high granularity, which we found to be unrealistically high for our application. Therefore, we tested several methods to solve a larger number of smaller systems. All methods show a significant speedup on test runs up to 512 processes. On the Touchstone Delta, the best efficiency achieved was about 40%.

Although problem sizes will significantly increase beyond the size of the problem studied in this paper, we believe that granularity will remain of the same order of magnitude or might even decrease significantly. For most problems, the total amount of arithmetic increases faster than linearly with the total amount of data (which itself is proportional to the number of unknowns). To keep the total execution time reasonable, one should use a number of nodes proportional to the total amount of arithmetic. Hence, the amount of data per node (the granularity) is likely to decrease. There is also a numerical reason why fine-grain concurrency will become increasingly important. To improve the convergence rate, one could consider

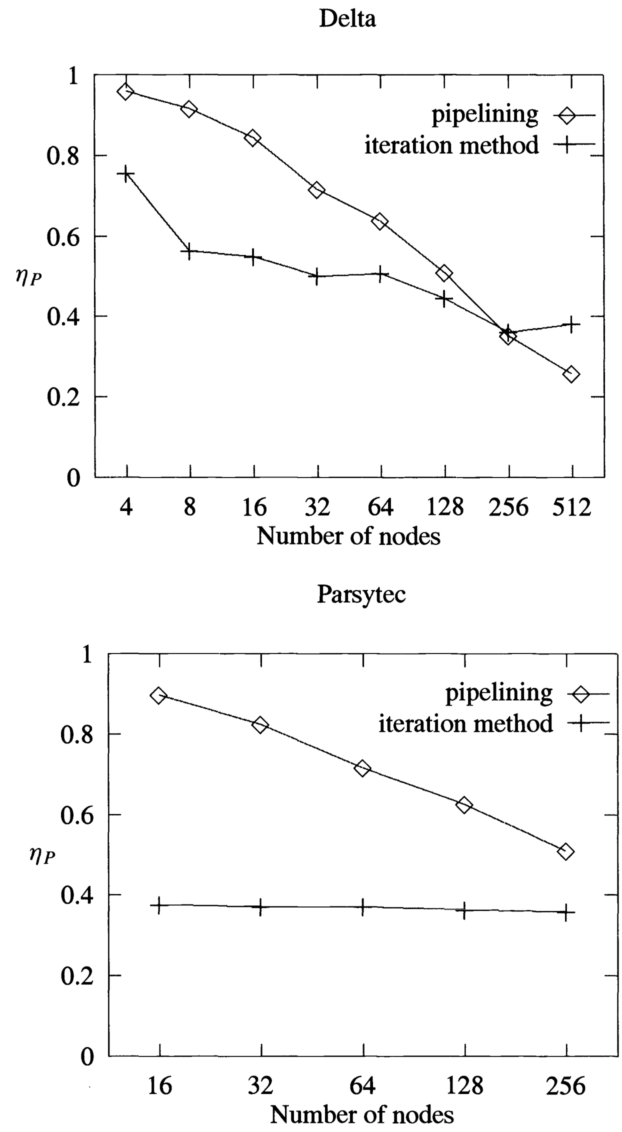


FIG. 17. *Efficiency of pipelining and iteration methods on the Delta and Parsytec multicomputers.*

using line-relaxation methods as smoothers in a nonlinear multigrid method. The coarser levels, however, quickly require computations of fine granularity.

Block-tridiagonal solvers that are efficient for fine-grain computations are difficult to implement. Nevertheless, it is possible to achieve significant execution-time improvements through the use of relatively fine-grain concurrency. A significant simplification for future endeavors is that our best results were obtained with the iterative block-tridiagonal solver, which is, from an algorithmic point of view, the simplest.

REFERENCES

[1] W. ALTHAUS, E. KRAUSE, J. HOFHAUS, AND M. WEIMER, *Bubble- and spiral-type breakdown of slender vortices*, Exp. Thermal Fluid Sci., (1995), to appear.

[2] S. BONDELI, *Divide and conquer: A parallel algorithm for the solution of a tridiagonal system of equations*, Parallel Comput., 17 (1991), pp. 419-434.

[3] M. BREUER, *Numerische Lösung der Navier-Stokes Gleichungen für dreidimensionale inkompressible instationäre Strömungen zur Simulation des Wirbelaufplatzens*, Ph.D. thesis, Aerodynamisches Institut der RWTH Aachen, Germany, 1991.

[4] M. BREUER AND D. HÄNEL, *A dual time-stepping method for 3-D, viscous, incompressible vortex flows*, Comput. & Fluids, 22 (1993), pp. 467-484.

[5] INTEL SUPERCOMPUTING SYSTEMS DIVISION, *Touchstone Delta System User's Guide*, Beaverton, Oregon, 1991.

[6] ———, *The Intel Paragon Supercomputer—an Overview*, Beaverton, Oregon, 1992.

[7] A. FROMMER, *Lösung linearer Gleichungssysteme auf Parallelrechnern*, Vieweg-Verlag, Braunschweig, Germany, 1990.

[8] C. HO AND S. JOHNSSON, *Optimizing tridiagonal solvers for alternating direction methods on Boolean cube multiprocessors*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 563-592.

[9] R. HOCKNEY, *A fast and direct solution of Poisson's equation using Fourier analysis*, J. Assoc. Comput. Mach., 1 (1965), pp. 95-113.

[10] R. HOCKNEY AND C. JESSHOPE, *Parallel Computers*, Adam Hilger, Bristol, 1981.

[11] E. ISAACSON AND H. KELLER, *Analysis of Numerical Methods*, John Wiley and Sons, New York, 1966.

[12] S. JOHNSSON, Y. SAAD, AND M. SCHULTZ, *Alternating direction methods on multiprocessors*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 686-700.

[13] W. JULING AND K. KREMER, *The massively parallel computer system of the DFG priority research program "flow simulation on supercomputers"*, in Notes on Numerical Fluid Mechanics, Vol. 38, Vieweg-Verlag, Braunschweig, Germany, 1993.

[14] A. KRECHEL, H.-J. PLUM, AND K. STÜBEN, *Parallelization and vectorization aspects of the solution of tridiagonal linear systems*, Parallel Comput., 14 (1990), pp. 31-49.

[15] H. STONE, *An efficient parallel algorithm for the solution of a tridiagonal linear system of equations*, J. Assoc. Comput. Mach., 20 (1973), pp. 27-38.

[16] H. WANG, *A parallel method for tridiagonal equations*, ACM Trans. Math. Software, 7 (1981), pp. 170-183.

SIAM J. SCI. COMPUT.
Vol. 17, No. 2, pp. 479-497, March 1996

RUNGE-KUTTA SOFTWARE WITH DEFECT CONTROL FOR BOUNDARY VALUE ODES*

W. H. ENRIGHT† AND P. H. MUIR‡

Abstract. A popular approach to the numerical solution of boundary value ODE problems involves the use of collocation methods. Such methods can be naturally implemented so as to provide a continuous approximation to the solution over the entire problem interval. On the other hand, several authors have suggested as an alternative, certain subclasses of the implicit Runge-Kutta formulas, known as mono-implicit Runge-Kutta (MIRK) formulas, which can be implemented at a lower cost per step than the collocation methods. These latter formulas do not have a natural implementation that provides a continuous approximation to the solution; rather, only a discrete approximation at certain points within the problem interval is obtained. However, recent work in the area of initial value problems has demonstrated the possibility of generating inexpensive interpolants for any explicit Runge-Kutta formula. These ideas have recently been extended to develop continuous extensions of the MIRK formulas. In this paper, we describe our investigation of the use of continuous MIRK formulas in the numerical solution of boundary value ODE problems. A primary thrust of this investigation is to consider defect control, based on the continuous MIRK formulas, as an alternative to the standard use of global error control, as the basis for termination and mesh redistribution criteria.

Key words. Runge-Kutta methods, boundary value ordinary differential equations, interpolants, defect control, numerical software

AMS subject classifications. 65L05, 65L10

1. Introduction.

A popular approach to the numerical solution of boundary value ODE (BVODE) problems involves the use of collocation formulas based on, for example, Gauss points. Gauss point formulas have been implemented in the widely used code, COLSYS [Ascher, Christiansen, and Russell, 1981] and its descendant, COLNEW [Bader and Ascher, 1987]. For many numerical methods for BVODE problems, one obtains an approximation to the solution at a discrete set of mesh points which partition the problem interval. An advantage of the collocation approach is that a continuous approximation to the solution over the entire problem interval is obtained as a natural consequence of the standard implementation of these methods. For a mesh with subintervals of size h and collocation based on Gauss points with s collocation points per subinterval, the continuous solution provides an approximation with global error $O(h^s)$, while the associated discrete solution at the mesh points has a global error that is (in the absence of order reduction phenomenon (see, e.g., [Prothero and Robinson, 1974]) $O(h^{2s})$, a property known as superconvergence. The continuous solution approximation can be very useful when the user requires solution information throughout the problem interval. Furthermore it can also be useful within the code, itself, for example, for error estimation, mesh refinement and redistribution, and generation of initial iterates for Newton iterations. In addition, the presence of a continuous approximate solution makes it possible to assess solution quality by examining the defect, which is the amount by which the approximate solution fails to satisfy the BVODE system.

It is well known (see, e.g., [Weiss, 1974]) that the collocation formulas, when applied to first-order systems of ODEs, can be viewed as equivalent to certain subclasses of implicit Runge-Kutta formulas. Recently several authors have suggested, as an alternative to the collocation formulas, other subclasses of implicit Runge-Kutta formulas for use in the numerical

*Received by the editors July 2, 1993; accepted for publication (in revised form) October 12, 1994. This work was supported by the Natural Sciences and Engineering Research Council of Canada and the Information Technology Research Center of Ontario.

†Department of Computer Science, University of Toronto, Toronto, Ontario, Canada M5S 1A4 (enright@na.toronto.edu).

‡Mathematics and Computer Science Department, St. Mary's University, Halifax, Nova Scotia, Canada B3H 3C3 (muir@stmarys.ca).

solution of BVODE problems. These formulas are known as mono-implicit Runge-Kutta (MIRK) formulas [van Bokhoven, 1980], [Cash and Singhal, 1982a]. A recent investigation of MIRK formulas is presented in [Burrage, Chipman, and Muir, 1994], where families of orders one through six are characterized and an order barrier result is presented. An advantage of these formulas over the collocation formulas is that the calculations on each subinterval, which use MIRK formulas in the setup of the Newton systems, are explicit and therefore can be implemented more efficiently [Cash and Singhal, 1982b], [Gupta, 1985], [Enright and Muir, 1986], [Cash, 1986, 1988]. The most recent work using these formulas [Cash and Wright, 1990, 1991] makes use of three specific MIRK formulas of orders four, six, and eight within a software package for the numerical solution of BVODEs, called HAGRON. An early software effort, described in [Gupta, 1985], involved modifying the PASVA code [Lentini and Pereyra, 1977] to use MIRK formulas. Both of these codes use deferred correction to provide error estimation and return a discrete solution approximation.

Unlike the collocation formulas, the MIRK formulas do not directly provide a continuous approximation to the solution. The idea of extending the discrete solution approximation to a continuous one has received considerable attention over the last few years, primarily in the area of initial value ODE problems. In that context, a number of authors have demonstrated the possibility of generating inexpensive interpolants for general explicit Runge-Kutta formulas; see, for example, [Enright, Jackson, Nørsett, and Thomsen, 1986], [Gladwell, Shampine, Baca, and Brankin, 1987], [Owren and Zennaro, 1991, 1992], and [Verner, 1993]. These interpolants are obtained by performing extra derivative evaluations, within the current step, thus preserving the one-step nature of the formula. Recently, [Muir and Owren, 1993] have developed continuous versions of the MIRK formulas. In addition to presenting a general analysis, they establish order barriers for continuous MIRK (CMIRK) schemes of orders one through six, and provide characterizations of CMIRK families of these orders. For BVODEs, the question of extending a discrete solution to obtain a continuous one has received less attention. Most BVODE codes do not provide a continuous solution approximation. Examples of such codes are the PASVA code and the modified version mentioned above, some experimental codes based on special one-sided Runge-Kutta schemes [Kreiss, Nichols, and Brown, 1986], [Brown and Lorenz, 1987], and the HAGRON code mentioned earlier.

In this paper, we investigate the computation of continuous solution approximations for the numerical solution of BVODEs based on CMIRK schemes. A primary thrust of the investigation involves the use of defect control as an alternative to the standard global error estimate control, for use in both mesh selection and accuracy control. The size of the defect gives a different measure of the suitability of the approximate solution; it is the amount by which the computed solution fails to satisfy the system of differential equations. It is a natural choice for control since it is the defect which arises in the analysis of the mathematical conditioning of the underlying problem where appropriate condition numbers are introduced to quantify the sensitivity of the global error to perturbations of the ODEs. See [Ascher, Mattheij, and Russell, 1988, Chap. 3, §4]. It has been suggested (see, e.g., [Cash and Silva, 1993]) that monitoring the defect may be appropriate in situations where difficulties arise in estimating the global error. The use of the defect has also been studied in the context of the numerical solution of initial value ODE problems; see, e.g., [Hanson and Enright, 1983], [Enright, 1989a, 1989b], and [Higham, 1989a, 1989b, 1991].

This paper is organized as follows. In §2 we describe the use of MIRK and CMIRK schemes in an algorithm for the numerical solution of BVODEs. In §3 we briefly discuss the class of CMIRK formulas and describe the specific continuous formulas we have chosen to implement. In §4, we identify several implementation issues associated with the development of software based on the algorithm of §2. In §5, we demonstrate the effectiveness of the software we have written by applying to some systems of first-order ODEs with uniform tolerance constraints. We also provide corresponding results from the COLNEW code for the same test set in order to show that the performance of our software is reasonable. It is not our intention to conduct extensive comparisons of the two codes. The COLNEW code has well-known advantages, e.g., it can handle mixed-order BVODE systems with different tolerance constraints for each component and multipoint conditions. We close, in §6, with our conclusions and a summary of our plans for future work in this area.

2. The numerical solution of BVODEs using MIRK and CMIRK schemes. In this section we review an algorithm for solving a system of BVODEs using MIRK and CMIRK schemes. The basic approach is to use the MIRK formulas to form a discrete algebraic system which can then be solved with a Newton iteration to obtain a discrete solution. Once this solution is obtained, a CMIRK scheme is employed to provide a continuous solution approximation over the problem interval for use in the computation of defect estimates, mesh redistribution, and initial guesses for subsequent Newton iterates.

We assume that the boundary value ODEs to be solved can be expressed in the general form,

$$y'(t) = f(t, y(t)), \qquad t \in [a, b],$$

where $y \in R^n$ and $f : R \times R^n \rightarrow R^n$, and where we assume for convenience of implementation that we have separated boundary conditions,

$$g(y(a), y(b)) = \begin{pmatrix} g_0(y(a)) \\ g_1(y(b)) \end{pmatrix} = 0,$$

where $g_0 : R^n \rightarrow R^{n_0}$ and $g_1 : R^n \rightarrow R^{n_1}$ and $n_0 + n_1 = n$.

A standard approach for employing a Runge-Kutta scheme to solve such problems can be described in terms of a two-level iteration scheme:

(0) Prior to beginning the two-level iteration we determine an initial mesh, $\{t_i\}_{i=0}^{N}$, which partitions $[a, b]$ with $t_0 = a$ and $t_N = b$, and an initial discrete solution approximation, $Y^{(0)} = [y_0^{(0)}, y_1^{(0)}, \ldots, y_N^{(0)}]^T$, where $y_i^{(0)} \approx y(t_i)$, the exact solution evaluated at t_i.

(1) The first step of the upper-level iteration is the setup and solution of a discrete system, $\Phi(Y) = 0$, where the "residual function"

$$\Phi(Y) = [g_0(y_0), \phi_1, \phi_2, \ldots, \phi_{N-1}, g_1(y_N)]^T.$$

Each vector ϕ_i is of size n and is defined by a Runge-Kutta scheme. We solve this discrete system using a Newton iteration, which constitutes the lower-level iteration. For $m = 0, 1, 2, \ldots,$ the Newton system has the form

$$\begin{pmatrix} \frac{\partial g_0}{\partial y_0}^{(m)} & 0 & 0 & \ldots & 0 & 0 \\ \frac{\partial \phi_0}{\partial y_0}^{(m)} & \frac{\partial \phi_0}{\partial y_1}^{(m)} & 0 & \ldots & 0 & 0 \\ 0 & \ddots & \ddots & \ldots & 0 & 0 \\ 0 & 0 & 0 & \ldots & \frac{\partial \phi_{N-1}}{\partial y_{N-1}}^{(m)} & \frac{\partial \phi_{N-1}}{\partial y_N}^{(m)} \\ 0 & 0 & 0 & \ldots & 0 & \frac{\partial g_1}{\partial y_N}^{(m)} \end{pmatrix} \begin{pmatrix} \Delta y_0^{(m)} \\ \vdots \\ \Delta y_N^{(m)} \end{pmatrix} = - \begin{pmatrix} g_0(y_0^{(m)}) \\ \phi_0^{(m)} \\ \vdots \\ \phi_{N-1}^{(m)} \\ g_1(y_N^{(m)}) \end{pmatrix},$$

and $y_i^{(m+1)} = y_i^{(m)} + \Delta y_i^{(m)}, i = 0, \ldots, N$. Here $y_i^{(m)}$ is the mth iterate value for y_i, $\Delta y_i^{(m)}$ is the corresponding Newton correction, $\phi_i^{(m)}$, $\frac{\partial \phi_i}{\partial y_i}^{(m)}$, and $\frac{\partial \phi_i}{\partial y_{i+1}}^{(m)}$ are the ith residual function component and its derivatives evaluated at $y_i^{(m)}$ and $y_{i+1}^{(m)}$, and $g_0(y_0^{(m)})$, $g_1(y_N^{(m)})$, $\frac{\partial g_0}{\partial y_0}^{(m)}$, and $\frac{\partial g_1}{\partial y_N}^{(m)}$ are the boundary conditions and derivatives evaluated at $y_0^{(m)}$ and $y_N^{(m)}$.

When a MIRK scheme is used as the underlying discretization the ith component of the residual function takes the form

$$\phi_i = y_{i+1} - y_i - h_i \sum_{r=1}^{s} b_r K_r, \tag{2.1}$$

where the stages, K_r, are given by

$$K_r = f\left(t_i + c_r h_i, (1 - v_r) y_i + v_r y_{i+1} + h_i \sum_{j=1}^{r-1} x_{rj} K_j\right), \qquad r = 1, \ldots, s. \tag{2.2}$$

Because the stages are defined explicitly in terms of the unknowns y_i and y_{i+1}, these methods can be implemented quite efficiently (see, e.g., [Cash, 1986]). The corresponding partial derivatives are also easily computed. We get

$$\frac{\partial \phi_i}{\partial y_i} = -I - h_i \sum_{r=1}^{s} b_r \frac{\partial K_r}{\partial y_i} \quad \text{and} \quad \frac{\partial \phi_i}{\partial y_{i+1}} = I - h_i \sum_{r=1}^{s} b_r \frac{\partial K_r}{\partial y_{i+1}}, \tag{2.3}$$

where, for $r = 1, \ldots, s$,

$$\frac{\partial K_r}{\partial y_i} = \mathbf{J}_r \left((1 - v_r) I + h_i \sum_{j=1}^{r-1} x_{rj} \frac{\partial K_j}{\partial y_i}\right) \text{ and } \frac{\partial K_r}{\partial y_{i+1}} = \mathbf{J}_r \left(v_r I + h_i \sum_{j=1}^{r-1} x_{rj} \frac{\partial K_j}{\partial y_{i+1}}\right). \tag{2.4}$$

Here $\mathbf{J}_r$ is the evaluation of $\frac{\partial f(t,y(t))}{\partial y}$ at the argument of the rth stage of the MIRK scheme, as in (2.2). If the Newton iteration does not converge, we go to step (5), with a new mesh that is obtained by halving each subinterval of the current mesh, and with the same current solution approximation.

(2) The converged Newton iteration yields a discrete solution approximation, $\{y_i\}_{i=0}^{N}$, for the given mesh. We next use an associated CMIRK scheme to construct a C^1 continuous solution approximation over the entire problem interval. The CMIRK scheme will usually employ the stages (2.2) of the MIRK scheme as well as a few additional stages of the same form as those in (2.2). It is important to note that the evaluation of these additional stages involves only explicit calculations and these are only performed after the Newton iteration has converged. As a result, the cost of obtaining the required continuous approximation is relatively small. On each subinterval the CMIRK scheme provides a polynomial interpolant to the approximate solution over that subinterval. The interpolant is obtained using information that is local to the step. (This is particularly relevant for a parallel implementation where it is important that the calculations for each subinterval be based only on local information.) The continuous solution approximation, $u(t)$, has the same order of accuracy as the underlying discrete solution. The defect, $\delta(t)$, is then given by

$$\delta(t) = u'(t) - f(t, u(t)), \quad a \le t \le b.$$

We sample the defect on each subinterval to obtain an estimate of its magnitude on that subinterval and terminate if this estimate is less than a given user-defined tolerance.

(3) If the termination condition is not met, the relative sizes of the maximum defect estimates associated with each subinterval are examined in the mesh selection algorithm to determine a more appropriate mesh. The basic idea is to redistribute the mesh while possibly adding or deleting points so that the maximum defect estimates obtained on the new mesh will be approximately the same on each subinterval (equidistribution of the defect) and will be bounded by the user tolerance. The maximum defect estimates are employed within a monitor function which provides the basis for the equidistribution process. Based on these considerations a new mesh is determined by employing standard strategies such as described in [Ascher, Mattheij, and Russell, 1988, Chap. 9, §1]. The algorithm is terminated, unsuccessfully, if the predicted number of mesh points for the new mesh is too large (according to a user-defined storage restriction).

(4) When a new mesh is determined, the continuous solution approximation is used to compute an initial discrete solution approximation for the next discrete problem and associated Newton iteration.

(5) This algorithm is then repeated, beginning at step (1), using the new mesh and solution approximation.

3. CMIRK schemes.

3.1. Overview. In the paper [Muir and Owren, 1993], the class of CMIRK schemes is investigated and we refer the reader to that paper for a description of these schemes. For the ith subinterval, the basic form of a CMIRK scheme is a polynomial in θ (where $\theta = (t - t_i)/h_i$),

$$u(t) = u(t_i + \theta h_i) = y_i + h_i \sum_{r=1}^{s^*} b_r(\theta) K_r, \quad 0 \le \theta \le 1, \quad t_i \le t \le t_{i+1}, \tag{3.1}$$

with s^* stages, K_r, of the same general form as in (2.2). (Usually the first s stages are those of an associated MIRK scheme. In such a case, the MIRK scheme is said to be embedded within the CMIRK scheme.) The main difference between MIRK schemes and CMIRK schemes is that the weight coefficients, $\{b_r\}_{r=1}^{s}$, of the MIRK scheme are replaced by weight polynomials, $\{b_r(\theta)\}_{r=1}^{s^*}$, in the CMIRK scheme.

In [Muir and Owren, 1993] multiparameter families of CMIRK formulas having a minimal number of stages for each order $p = 1, \ldots, 6$ are presented. Similarly, [Burrage, Chipman and Muir, 1994] identifies multiparameter families of discrete MIRK schemes having a minimal number of stages for each order $p = 1, \ldots, 6$. We will employ in our software the standard symmetric MIRK schemes of orders four and six. These two schemes are included in the families presented in [Burrage, Chipman, and Muir, 1994] and were first mentioned in the literature in [van Bokhoven, 1980]. We choose symmetric schemes since this can be advantageous (see [Ascher, Mattheij, and Russell, 1988, p. 440]). Since it is efficient to reuse the stages from the MIRK scheme within the CMIRK scheme, we will embed the MIRK schemes within optimal CMIRK schemes from [Muir and Owren, 1993], when possible.

In the remainder of this section we will present fourth-order and sixth-order CMIRK schemes containing optimal embedded MIRK schemes. These representative schemes will give us the opportunity to assess the role that the order of the MIRK and CMIRK schemes plays in the overall performance of a BVODE code. We are currently extending the results of [Burrage, Chipman, and Muir, 1994] and [Muir and Owren, 1993] to develop eighth-order MIRK and CMIRK families. An eighth-order MIRK scheme has already appeared in the literature (see, e.g., [Gupta, 1985]).

The coefficients defining a CMIRK scheme are usually presented in the form of a tableau as in Table 3.1.

TABLE 3.1

c_1	v_1	0	0	$\dots$	0	0
c_2	v_2	x_{21}	0	$\dots$	0	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
c_{s-1}	v_{s-1}	$x_{s-1,1}$	$x_{s-1,2}$	$\dots$	0	0
c_s	v_s	$x_{s,1}$	$x_{s,2}$	$\dots$	$x_{s,s-1}$	0
		$b_1(\theta)$	$b_2(\theta)$	$\dots$	$b_{s-1}(\theta)$	$b_s(\theta)$

TABLE 3.2
A four-stage fourth-order CMIRK scheme.

0	0	0	0	0	0
1	1	0	0	0	0
$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{8}$	$\frac{-1}{8}$	0	0
$\frac{3}{4}$	$\frac{27}{32}$	$\frac{3}{64}$	$\frac{-9}{64}$	0	0
		$b_1(\theta)$	$b_2(\theta)$	$b_3(\theta)$	$b_4(\theta)$

3.2. CMIRK schemes of fourth order. The general form for the family of fourth-order CMIRK schemes is given in [Muir and Owren, 1993]. It uses four stages and has free parameters c_3, c_4, and v_4. It is possible to choose c_3 so that the unique optimal, three-stage, fourth-order discrete MIRK scheme ([van Bokhoven, 1980], Method VI) is embedded within this CMIRK family. The first three stages of the CMIRK scheme will be the same as those of this MIRK scheme and the values of the first three continuous weight polynomials at $\theta = 1$ will be the weights of the discrete scheme.

A careful analysis to determine suitable optimization criteria based, for example, on minimizing the error coefficients of the dominant error terms followed by a thorough numerical investigation would be needed to determine optimal values for the remaining free parameters, c_4 and v_4. However, experience with the selection of such parameters for explicit Runge-Kutta pairs (see, e.g., [Sharp and Smart, 1993]) has indicated that as long as one avoids "pathological choices" for these parameters, the overall performance of a code will not be particularly sensitive to the choice of these parameters. After some preliminary testing to establish that the choices lead to reasonable schemes, we chose $c_4 = \frac{3}{4}$ and $v_4 = \frac{27}{32}$. This gives the fourth-order CMIRK scheme, given in Table 3.2, where

$$b_1(\theta) = -\frac{\theta\,(2\theta-3)\left(2\theta^2-3\theta+2\right)}{6}, \quad b_2(\theta) = \frac{\theta^2\left(9-20\theta+12\theta^2\right)}{6},$$

$$b_3(\theta) = \frac{2\theta^2\left(9-14\theta+6\theta^2\right)}{3}, \quad b_4(\theta) = -\frac{16\theta^2\,(\theta-1)^2}{3}.$$

3.3. CMIRK schemes of sixth order. A particular family of eight-stage, sixth-order CMIRK schemes is given in [Muir and Owren, 1993]. Although the family has nine free parameters $c_3, c_4, c_5, c_6, c_7, v_7, c_8, v_8, x_{81}$, it is not possible to embed within it the standard five-stage, sixth-order discrete MIRK schemes ([van Bokhoven, 1980], Method VIII). In order to get a family of sixth-order CMIRK schemes within which the optimal five-stage, sixth-order discrete scheme of [van Bokhoven, 1980] can be embedded, it is necessary to look outside the sixth-order CMIRK family in [Muir and Owren, 1993] and derive a new sixth-order CMIRK family. An analysis of the continuous order conditions up to order six shows that

TABLE 3.3
A nine-stage sixth-order CMIRK scheme.

0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0
$\frac{1}{4}$	$\frac{5}{32}$	$\frac{9}{64}$	$\frac{-3}{64}$	0	0	0	0	0	0	0
$\frac{3}{4}$	$\frac{27}{32}$	$\frac{3}{64}$	$\frac{-9}{64}$	0	0	0	0	0	0	0
$\frac{1}{2}$	$\frac{1}{2}$	$\frac{-5}{24}$	$\frac{5}{24}$	$\frac{2}{3}$	$\frac{-2}{3}$	0	0	0	0	0
$\frac{7}{16}$	$\frac{7}{16}$	$\frac{1547}{32768}$	$\frac{-1225}{32768}$	$\frac{749}{4096}$	$\frac{-287}{2048}$	$\frac{-861}{16384}$	0	0	0	0
$\frac{1}{8}$	$\frac{1}{8}$	$\frac{83}{1536}$	$\frac{-13}{384}$	$\frac{283}{1536}$	$\frac{-167}{1536}$	$\frac{-49}{512}$	0	0	0	0
$\frac{9}{16}$	$\frac{9}{16}$	$\frac{1225}{32768}$	$\frac{-1547}{32768}$	$\frac{287}{2048}$	$\frac{-749}{4096}$	$\frac{861}{16384}$	0	0	0	0
$\frac{3}{8}$	$\frac{3}{8}$	$\frac{233}{3456}$	$\frac{-19}{1152}$	0	0	0	$\frac{-5}{72}$	$\frac{7}{72}$	$\frac{-17}{216}$	0
		$b_1(\theta)$	$b_2(\theta)$	$b_3(\theta)$	$b_4(\theta)$	$b_5(\theta)$	$b_6(\theta)$	$b_7(\theta)$	$b_8(\theta)$	$b_9(\theta)$

such a scheme may require one additional stage beyond that required by an optimal continuous CMIRK scheme of sixth-order. Hence we have derived a nine-stage, sixth-order CMIRK family with an optimal embedded five-stage, sixth-order MIRK scheme, which we present in Table 3.3. The free parameters have been chosen to avoid "pathological cases," as was done for the fourth-order case. The first five stages of the CMIRK scheme are the stages of this MIRK scheme. The first five weight polynomials evaluated at $\theta = 1$ are the weights of this MIRK scheme. Recall, from §2, that the Newton iteration will employ only the five-stage discrete MIRK scheme and only after the Newton iteration converges will the extra four stages be evaluated in order to determine the CMIRK scheme.

The weight polynomials in Table 3.3 are

$$b_1(\theta) = \frac{\theta\left(334530 - 1287315\,\theta - 1662100\,\theta^2 + 10043400\,\theta^3 - 11467776\,\theta^4 + 4065280\,\theta^5\right)}{334530},$$

$$b_2(\theta) = \frac{\theta^2\left(3083913 - 25341580\,\theta + 62657400\,\theta^2 - 62874624\,\theta^3 + 22657024\,\theta^4\right)}{2341710},$$

$$b_3(\theta) = -\frac{16\,\theta^2\left(8505 - 69430\,\theta + 168780\,\theta^2 - 164352\,\theta^3 + 56320\,\theta^4\right)}{7965},$$

$$b_4(\theta) = -\frac{16\,\theta^2\left(8505 - 69430\,\theta + 168780\,\theta^2 - 164352\,\theta^3 + 56320\,\theta^4\right)}{7965},$$

$$b_5(\theta) = -\frac{2\,\theta^2\left(8505 - 69430\,\theta + 168780\,\theta^2 - 164352\,\theta^3 + 56320\,\theta^4\right)}{2655},$$

$$b_6(\theta) = -\frac{1024\,\theta^2\left(45568\,\theta^2 - 45856\,\theta + 7533\right)(\theta - 1)^2}{33453},$$

$$b_7(\theta) = \frac{128\,\theta^2\left(128\,\theta^2 - 128\,\theta + 21\right)(\theta - 1)^2}{15},$$

$$b_8(\theta) = \frac{1024\,\theta^2\left(134144\,\theta^2 - 132128\,\theta + 21609\right)(\theta - 1)^2}{234171}, \quad b_9(\theta) = -\frac{16384\,\theta^3(\theta - 1)^3}{441}.$$

4. Implementation issues. In this section we comment on several of the algorithms and heuristics employed within the software we have written for the numerical solution of BVODEs, based on continuous MIRK schemes and defect control, for which the general algorithm was given in §2. The reader is referred to [Enright and Muir, 1993] for further details related to this discussion.

4.1. The modified Newton iteration. To solve the system, $\Phi(Y) = 0$, described in §2, we use a combination of a damped Newton iteration and a fixed Jacobian iteration, with a scheme for switching between the two. The switching scheme involves monitoring the rate of decrease of a function known as the "natural criterion function," $||J(Y^{(m)})^{-1}\Phi(Y^{(m+1)})||$, (see, e.g., [Ascher, Mattheij, and Russell, 1988, Chap. 8]), where $Y^{(m)}$ is the mth iterate, $\Phi(Y)$ is the residual function defined in §2, and $J(Y)$ is the Jacobian of $\Phi(Y)$.

A point of some concern here is the evaluation of the blocks of the Newton matrix, as given in (2.3). The literature for initial value ODEs (see, e.g., [Butcher 1987, p. 214]) indicates that the usual practice is to evaluate $\frac{\partial f}{\partial y}$ at only one point on each subinterval, with this one value being used in place of each instance of J_r arising in (2.4). For boundary value ODEs, [Cash, 1986, p. 1037] describes an approach which involves evaluating $\frac{\partial f}{\partial y}$ once for each subinterval. This same idea is mentioned in [Ascher, Mattheij, and Russell, 1988, p. 217]. On the other hand, the COLNEW code performs evaluations of $\frac{\partial f}{\partial y}$ at each of the s collocation points of each subinterval. We implemented both approaches, i.e., a scheme with one evaluation per subinterval and a scheme with s evaluations per subinterval, as in (2.4). We found major differences in performance between the two approaches, especially when applied to difficult nonlinear problems. When using the single Jacobian evaluation per subinterval approach, we found some savings in the cost of computing the Jacobian blocks, but the resulting degradation in the performance of the Newton iteration more than offset these savings. Not only was the rate of convergence for the single Jacobian evaluation approach much slower, but also the likelihood of achieving convergence was much lower. This suggests that on some problems it is better to incur the extra cost of using multiple Jacobian values on each subinterval in order to achieve a lower overall cost, and we have adopted this approach.

4.2. Defect estimation. Both the termination criterion and the mesh selection algorithms we employ require an estimate of the maximum value of the defect on each subinterval, as discussed in §2. To avoid excessive computational costs, it is important to obtain an estimate of these maximums by sampling the defect at as few points as possible. In an analogous setting, COLNEW samples the global error at two points per subinterval but it is indicated in [Ascher and Jacobs, 1989] that sampling at the s collocation points of each subinterval can be advantageous. In the initial value ODE setting, defect estimation based on one sample point per step has been found to be quite reliable (see, e.g., [Enright, 1989b]). Employing a standard analysis for continuous Runge-Kutta schemes, we determined suitable candidates for the defect sample points, $0 \leq \theta^* \leq 1$, within each subinterval, for the two CMIRK schemes of §3. For the fourth-order CMIRK scheme we choose $\theta^* \approx 0.226$, while for the sixth-order CMIRK scheme we choose $\theta^* \approx 0.716$.

We performed numerical experiments to determine the reliability of one-point sampling and also two-point sampling, with the second sample point chosen to be $1 - \theta^*$ (due to the presence of certain symmetries in the form of the defect). We compared the estimates for each subinterval with a "true" maximum defect obtained by sampling the defect at many points on the subinterval, and monitored the number of times that the estimation schemes were deceived, i.e., reported a maximum defect that was less than 20% of the "true" maximum defect. We found that sampling at two points provided substantially more reliability; the maximum percentage of deceptions reported by the two-point sampling scheme was usually an order of magnitude smaller than that of the one-point sampling scheme, and was generally 1% or lower. Furthermore, although the execution time specifically associated with the defect estimation module was approximately doubled when two-point sampling was used, the overall execution cost was approximately the same. This occurs because the availability of a sharper estimate of the defect profile over the mesh subintervals allows the mesh redistribution algorithm to determine meshes that are better suited to the underlying solution behavior, thus leading to a final acceptable solution more quickly.

4.3. Mesh selection. We refer the reader to [Ascher, Mattheij, and Russell, 1988, Chap. 9] for an overview of a generic mesh selection algorithm for BVODE codes. The mesh selection algorithm we have implemented is based on this model algorithm but uses defect estimates rather than global error estimates as the basis for the equidistribution process. There are several parameters that control various aspects of the performance of a mesh selection algorithm, such as the frequency of application of the refinement and redistribution process and the estimation of the number of points needed for the new mesh. We conducted a number of experiments to examine several of these parameters and have reported the results in [Enright and Muir, 1993]. One result concerns the parameter which determines when the redistribution is applied. In COLNEW, the value of this parameter is chosen so that mesh redistribution is used only when the maximum error estimate over all subintervals is much larger than the average value, and as a result, mesh redistributions within COLNEW are infrequent. We have profiled our software and have noted that the execution costs associated with the mesh selection algorithm are relatively small, so efficiency concerns over execution costs directly associated with the mesh selection module are not an issue. Further testing has indicated that substantial savings in overall execution costs are usually obtained when mesh redistribution is employed, and thus our software invokes its mesh redistribution algorithm for every converged solution. This makes it possible to adapt the mesh to the approximate solution more quickly than when mesh doubling is used.

It should be noted that the defect estimate can be computed and the termination criterion applied whenever a converged solution is obtained on a given mesh. This can be contrasted with the scheme used in COLNEW where termination occurs only when the current solution can be compared with the previously computed solution obtained on a mesh which is exactly half the current one. (That is, the current mesh is obtained by halving each subinterval of the previous mesh.) This appears to have influenced the choice within COLNEW to favor mesh doubling over mesh redistribution.

5. Performance analysis.

5.1. Overview. In this section we present some results on the performance of our BVODE software, called MIRKDC, which uses MIRK and CMIRK schemes and defect control, as described in the previous sections. We will present results for 18 test problems selected from three problem families. In order to demonstrate that the performance of MIRKDC on this test set is reasonable, we will also present the results of applying COLNEW to this same test set. It is not the purpose of this paper to conduct a comparison of these codes. See [Pereyra and Russell, 1982] for a discussion of the difficulties involved in such comparisons. The purpose of presenting these results is to show that the performance of MIRKDC is reasonable, compared to that of a well-known, widely used package, for a particular test set. It is essential to have both codes treat the same problem class and to apply the tolerance request in the same way. Thus we consider the solution of first-order systems with the same tolerance applied to all solution and derivative components since this is the problem class that MIRKDC can currently handle. COLNEW has more flexibility; e.g., it can handle mixed-order multipoint ODE systems directly and can apply different tolerances to each solution and derivative component.

We shall quantify the performance according to several criteria. For each test problem, we will first present a "performance profile." This traces the sequence of discrete problems that a code attempts during its efforts to compute a final solution. The performance profile consists of a list of ordered pairs, (n, q), one pair for each discrete system, where n is the number of subintervals in the mesh from which the discrete system was derived, and q is the number of Newton iterations performed on that discrete problem. The profile gives an indication of the costs incurred by the code in attempting to solve the given problem. In addition to the profile, our assessment of the performance of each code on a given problem will include the CPU time (in seconds, on a DEC Micro-Vax 3800), the global error of the computed solution, and the defect of the solution. The global error is computed by comparing the computed solution with a high accuracy numerical solution at one hundred points uniformly distributed over the problem interval.

An additional difficulty here is that COLNEW controls the global error while MIRKDC controls the defect. These two measures of solution quality are not equivalent (although they are related as mentioned earlier and as given in [Ascher, Mattheij, and Russell, 1988, Chap. 3, §4]). COLNEW is not designed to control the defect and would require modification in order to allow it to compute a continuous solution approximation appropriate for use in a defect control setting. On the other hand, the global error of the continuous solution produced by MIRKDC can be measured, even though this quantity is not controlled by the code. Thus we arrange for each code to return a solution for which the global error on each solution and derivative component is within a given tolerance. Since COLNEW is designed to control the global error, it is straightforward to make this specification to COLNEW. MIRKDC controls the defect so, for each test problem, we experimentally determine a tolerance for the defect that will yield a solution whose global error is within the desired tolerance.

COLNEW can handle mixed-order BVODE systems directly and this is often an advantage over all codes that can treat only first-order systems. Since MIRKDC can only handle first-order systems, our set of test problems will be presented as first-order systems. However there are possibilities for improving the flexibility of MIRK-based BVODE solvers. When we consider a BVODE system obtained from the conversion of a mixed-order system to a first-order system, we note that the matrices that arise during the setup of the almost block diagonal system do exhibit some structure which could be exploited by a special purpose algorithm. We are currently investigating matrix setup algorithms which will take advantage of this special structure. Another approach we are also investigating involves developing extensions of the MIRK and CMIRK formulas for the direct treatment of higher-order ODE systems (analogous to the situation for initial value ODE problems where Runge-Kutta-Nystrom formulas are available).

For each test problem, we require each code to compute two solutions, one with a global error of less than 10^{-6} and one with a global error of less than 10^{-8}. For MIRKDC, for each test problem, we give the tolerance value that was imposed on the defect in order to obtain a solution with the desired global error. We provide results for both fourth- and sixth-order methods. A "*" indicates that the Newton iteration for a discrete problem did not converge. A failure is indicated (by the entry $FAIL$) if a code required more than 2000 subintervals to solve the problem. In all cases where failure occurred, the performance profile was always $(10, *)$, $(20, *)$, $(40, *)$, $(80, *)$, $(160, *)$, $(320, *)$, $(640, *)$, $(1280, *)$, so we do not list it in the tables.

We make the observation that all the test problems we consider here are single problem instances, although they are chosen from problem families. In particular, no use is made of "parameter continuation," in which the solution and mesh from one problem are used as the initial solution guess and initial mesh for a more difficult problem from the same family.

TABLE 5.1
COLNEW on Problem 1: *fourth order,* tol $= 10^{-6}$.

$\epsilon \Rightarrow$	1.0	0.1	0.05	0.01	0.005	0.001	0.0005
Profile	(5, 2)	(5, 2)	(5, 3)	(5, ∗)(116, 1)	(5, ∗)(71, 1)	(5, ∗)(40, 1)	(5, ∗)(80, 1)
	(10, 1)	(10, 1)	(10, 1)	(10, 5)	(10, 6)(142, 1)	(10, 9)(80, 1)	(10, ∗)(160, 1)
		(20, 1)	(20, 1)	(20, 1)	(10, 1)	(10, 2)(160, 1)	(20, ∗)(110, 1)
		(40, 1)	(40, 1)	(40, 1)	(20, 1)	(20, 2)(320, 1)	(40, 10)(220, 1)
			(80, 1)	(80, 1)	(40, 1)	(20, 1)	(40, 1)(440, 1)
				(58, 1)	(80, 1)	(40, 1)	(40, 1)
Time	0.078	0.313	0.645	1.415	1.697	2.963	6.869
Error	$8.3 \cdot 10^{-7}$	$2.3 \cdot 10^{-7}$	$6.2 \cdot 10^{-8}$	$1.6 \cdot 10^{-7}$	$1.8 \cdot 10^{-7}$	$1.2 \cdot 10^{-7}$	$2.4 \cdot 10^{-8}$
Defect	$1.0 \cdot 10^{-4}$	$2.1 \cdot 10^{-4}$	$1.5 \cdot 10^{-4}$	$9.4 \cdot 10^{-4}$	$2.0 \cdot 10^{-3}$	$3.9 \cdot 10^{-3}$	$2.1 \cdot 10^{-3}$

TABLE 5.2
MIRKDC on Problem 1: *fourth order, error* $= 10^{-6}$.

$\epsilon \Rightarrow$	1.0	0.1	0.05	0.01	0.005	0.001	0.0005
tol $\Rightarrow$	$6 \cdot 10^{-7}$	$8 \cdot 10^{-6}$	$2 \cdot 10^{-6}$	$2 \cdot 10^{-6}$	$3 \cdot 10^{-6}$	$4 \cdot 10^{-6}$	$3 \cdot 10^{-7}$
Profile	(5, 1)	(5, 1)	(5, 1)	(5, 2)	(5, ∗)(125, 1)	(5, ∗)(153, 1)	(5, 5)(323, 1)
	(16, 1)	(20, 1)	(20, 1)	(20, 2)	(10, 2)	(10, 4)(175, 1)	(20, ∗)(373, 1)
	(21, 1)	(29, 1)	(45, 1)	(80, 1)	(40, 1)	(40, ∗)(192, 1)	(40, 5)
			52, 1)	(111, 1)	(104, 1)	(80, 4)	(160, 1)
Time	0.059	0.078	0.169	0.358	0.468	1.468	1.667
Error	$3.7 \cdot 10^{-7}$	$9.6 \cdot 10^{-7}$	$7.6 \cdot 10^{-7}$	$7.2 \cdot 10^{-7}$	$8.9 \cdot 10^{-7}$	$8.4 \cdot 10^{-7}$	$8.0 \cdot 10^{-7}$
Defect	$3.5 \cdot 10^{-7}$	$4.3 \cdot 10^{-6}$	$1.5 \cdot 10^{-6}$	$3.0 \cdot 10^{-6}$	$4.5 \cdot 10^{-6}$	$9.7 \cdot 10^{-6}$	$2.0 \cdot 10^{-6}$

This causes the performance of the Newton iteration to play a larger role in determining the execution time required by the code to solve a given problem. If the performance of a code is examined in a parameter continuation setting, one expects the initial guesses and meshes will be sufficiently good that the Newton iterations usually will converge more readily, and thus the costs of other parts of the codes will play a larger role in determining overall performance. Of course, the success of a parameter continuation scheme depends heavily on the strategy used to select the parameter steps; thus in our current study we avoid the use of parameter continuation, preferring to study code performance in the simpler setting where each problem is solved independently.

5.2. Numerical results. For each problem family, the initial guesses for the solution are chosen to be straight lines through the boundary conditions, where applicable, and zero otherwise.

The first family of test problems is the Swirling Flow III family given in [Ascher, Mattheij, and Russell, 1988, p. 23]. The problem statement given there, however, has a slight error. The corrected form of the problem, which can be found in, for example, [Batchelor, 1951], is

$$\epsilon f'''' + ff''' + gg' = 0, \qquad \epsilon g'' + fg' - f'g = 0,$$

with

$$f(0) = f(1) = f'(0) = f'(1) = 0, \quad g(0) = \Omega_0, \quad g(1) = \Omega_1.$$

For our testing, we chose $\Omega_0 = 1$ and $\Omega_1 = -1$ (which specifies a counter-rotating disk problem) and consider values of the viscosity parameter $\epsilon = 1.0, 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005$. The results are given in Tables 5.1-5.8.

TABLE 5.3
COLNEW on Problem 1*: fourth order,* tol $= 10^{-8}$.

$\epsilon \Rightarrow$	1.0	0.1	0.05	0.01	0.005	0.001	0.0005
Profile	(5, 2)	(5, 3)	(5, 3)	(5, ∗)(320, 1)	(5, ∗)(160, 1)	(5, ∗)(80, 1)	(5, ∗)(160, 1)
	(10, 1)	(10, 1)	(10, 1)	(10, 6)	(10, 7)(320, 1)	(10, 9)(160, 1)	(10, ∗)(160, 1)
	(20, 1)	(20, 1)	(20, 1)	(20, 1)	(10, 1)	(10, 2)(320, 1)	(20, ∗)(320, 1)
	(40, 1)	(40, 1)	(40, 1)	(40, 1)	(20, 1)	(20, 2)(640, 1)	(40, 10)(640, 1)
		(80, 1)	(80, 1)	(80, 1)	(40, 1)	(20, 1)(325, 1)	(40, 2)
		(160, 1)	(160, 1)	(80, 1)	(80, 1)	(40, 1)(650, 1)	(40, 2)
				(160, 1)	(80, 1)	(40, 1)	(80, 2)
Time	0.28	1.13	1.15	3.00	3.13	9.73	9.26
Error	$2.5 \cdot 10^{-9}$	$8.6 \cdot 10^{-10}$	$4.1 \cdot 10^{-9}$	$2.2 \cdot 10^{-9}$	$6.3 \cdot 10^{-9}$	$1.0 \cdot 10^{-9}$	$5.4 \cdot 10^{-9}$
Defect	$1.8 \cdot 10^{-6}$	$3.4 \cdot 10^{-6}$	$1.9 \cdot 10^{-5}$	$4.6 \cdot 10^{-5}$	$1.8 \cdot 10^{-4}$	$2.3 \cdot 10^{-4}$	$6.8 \cdot 10^{-4}$

TABLE 5.4
MIRKDC on Problem 1*: fourth order, error* $= 10^{-8}$.

$\epsilon \Rightarrow$	1.0	0.1	0.05	0.01	0.005	0.001	0.0005
tol $\Rightarrow$	$7 \cdot 10^{-9}$	$8 \cdot 10^{-8}$	$1 \cdot 10^{-8}$	$3 \cdot 10^{-8}$	$3 \cdot 10^{-8}$	$3 \cdot 10^{-8}$	$3 \cdot 10^{-9}$
Profile	(5, 1)	(5, 1)	(5, 1)	(5, 3)(323, 1)	(5, 5)(341, 1)	(5, ∗)(320, 1)	(5, 5)(640, 1)
	(20, 1)	(20, 1)	(20, 1)	(20, 2)	(10, 2)(397, 1)	(10, 4)(536, 1)	(20, ∗)(1071, 1)
	(52, 1)	(71, 1)	(80, 1)	(80, 1)	(40, 1)	(40, ∗)(595, 1)	(40, 5)(1186, 1)
		(92, 1)	(168, 1)	(256, 1)	(160, 1)	(80, 4)	(160, 1)
Time	0.106	0.259	0.393	1.000	1.390	2.821	4.852
Error	$1.0 \cdot 10^{-8}$	$9.5 \cdot 10^{-9}$	$7.0 \cdot 10^{-9}$	$9.9 \cdot 10^{-9}$	$9.1 \cdot 10^{-9}$	$9.0 \cdot 10^{-9}$	$9.1 \cdot 10^{-9}$
Defect	$9.3 \cdot 10^{-9}$	$5.3 \cdot 10^{-8}$	$1.3 \cdot 10^{-8}$	$2.7 \cdot 10^{-8}$	$2.4 \cdot 10^{-8}$	$2.8 \cdot 10^{-8}$	$3.4 \cdot 10^{-9}$

TABLE 5.5
COLNEW on Problem 1*: sixth order,* tol $= 10^{-6}$.

$\epsilon \Rightarrow$	1.0	0.1	0.05	0.01	0.005	0.001	0.0005
Profile	(5, 2)	(5, 3)	(5, 3)	(5, ∗)	(5, ∗)(80, 1)	(5, ∗)(20, 1)	(5, ∗)(20, 1)
	(10, 1)	(10, 1)	(10, 1)	(10, 5)	(10, 5)	(10, 9)(40, 1)	(10, 11)(40, 1)
			(20, 1)	(20, 1)	(20, 1)	(10, 1)(80, 1)	(10, 1)(80, 1)
				(40, 1)	(40, 1)	(20, 1)	(20, 1)
Time	0.178	0.183	0.410	1.053	1.875	2.557	2.646
Error	$2.2 \cdot 10^{-10}$	$9.6 \cdot 10^{-8}$	$1.1 \cdot 10^{-8}$	$3.6 \cdot 10^{-8}$	$5.0 \cdot 10^{-9}$	$2.3 \cdot 10^{-8}$	$1.3 \cdot 10^{-7}$
Defect	$4.3 \cdot 10^{-8}$	$1.3 \cdot 10^{-5}$	$6.4 \cdot 10^{-6}$	$9.4 \cdot 10^{-5}$	$3.8 \cdot 10^{-5}$	$4.1 \cdot 10^{-4}$	$2.7 \cdot 10^{-3}$

TABLE 5.6
MIRKDC on Problem 1*: sixth order, error* $= 10^{-6}$.

$\epsilon \Rightarrow$	1.0	0.1	0.05	0.01	0.005	0.001	0.0005
tol $\Rightarrow$	$1 \cdot 10^{-4}$	$6 \cdot 10^{-5}$	$1 \cdot 10^{-4}$	$1 \cdot 10^{-5}$	$9 \cdot 10^{-6}$	$5 \cdot 10^{-6}$	$7 \cdot 10^{-6}$
Profile	(5, 1)	(5, 1)	(5, 1)	(5, 2)(42, 1)	(5, 3)(49, 1)	(5, 14)(74, 1)	(5, 11)(82, 1)
		(10, 1)	(12, 1)	(20, 1)	(20, ∗)(53, 1)	(20, ∗)(82, 1)	(20, ∗)(90, 1)
		(12, 1)	(15, 1)	(38, 1)	(40, 3)	(40, 4)(90, 1)	(40, 5)
Time	0.015	0.074	0.090	0.310	0.933	1.232	1.118
Error	$6.1 \cdot 10^{-7}$	$7.0 \cdot 10^{-7}$	$9.5 \cdot 10^{-7}$	$5.7 \cdot 10^{-7}$	$6.6 \cdot 10^{-7}$	$4.9 \cdot 10^{-7}$	$9.4 \cdot 10^{-7}$
Defect	$4.5 \cdot 10^{-6}$	$1.1 \cdot 10^{-5}$	$2.6 \cdot 10^{-5}$	$2.6 \cdot 10^{-6}$	$1.8 \cdot 10^{-5}$	$3.7 \cdot 10^{-6}$	$8.1 \cdot 10^{-6}$

The second problem family is the Swirling Flow I family from [Ascher, Mattheij, and Russell, 1988, p. 22]. It includes the problem parameters L, the length of the problem interval, and γ, the Rossby number. For our testing, we chose $L = 10.0$ and values of $\gamma = 0.0$, 5.0, 10.0, 20.0, 50.0, 100.0. The results are given in Tables 5.9-5.16.

TABLE 5.7
COLNEW on Problem 1*: sixth order,* tol = 10^{-8}.

$\epsilon \Rightarrow$	1.0	0.1	0.05	0.01	0.005	0.001	0.0005
Profile	(5, 2)	(5, 3)	(5, 3)	(5, *)(80, 1)	(5, *)(80, 1)	(5, *)(20, 1)	(5, *)(20, 1)
	(10, 1)	(10, 1)	(10, 1)	(10, 6)(160, 1)	(10, 6)(160, 1)	(10, 10)(40, 1)	(10, 11)(40, 1)
		(20, 1)	(20, 1)	(20, 1)	(20, 1)	(10, 1)(80, 1)	(10, *)(80, 1)
			(40, 1)	(40, 1)	(40, 1)	(20, 1)(160, 1)	(20, 3)(160, 1)
Time	0.169	0.348	0.778	1.955	3.589	4.212	4.702
Error	$2.2 \cdot 10^{-10}$	$1.4 \cdot 10^{-9}$	$1.6 \cdot 10^{-10}$	$5.0 \cdot 10^{-10}$	$8.7 \cdot 10^{-11}$	$5.2 \cdot 10^{-10}$	$2.5 \cdot 10^{-9}$
Defect	$4.3 \cdot 10^{-8}$	$4.8 \cdot 10^{-7}$	$2.3 \cdot 10^{-7}$	$3.5 \cdot 10^{-6}$	$1.3 \cdot 10^{-6}$	$1.5 \cdot 10^{-5}$	$1.0 \cdot 10^{-4}$

TABLE 5.8
MIRKDC on Problem 1*: sixth order, error* = 10^{-8}.

$\epsilon \Rightarrow$	1.0	0.1	0.05	0.01	0.005	0.001	0.0005
tol $\Rightarrow$	$5 \cdot 10^{-7}$	$8 \cdot 10^{-7}$	$6 \cdot 10^{-7}$	$1 \cdot 10^{-7}$	$1 \cdot 10^{-7}$	$7 \cdot 10^{-8}$	$7 \cdot 10^{-8}$
Profile	(5, 1)	(5, 1)	(5, 1)	(5, 3)(87, 1)	(5, 3)(92, 1)	(5, 14)(135, 1)	(5, 11)(150, 1)
	(9, 1)	(18, 1)	(20, 1)	(20, 1)	(20, *)(104, 1)	(20, *)(162, 1)	(20, *)(183, 1)
		(23, 1)	(32, 1)	(72, 1)	(40, 3)	(40, 4)(178, 1)	(40, 5)(201, 1)
Time	0.039	0.125	0.161	0.541	1.178	1.886	2.160
Error	$6.9 \cdot 10^{-9}$	$7.2 \cdot 10^{-9}$	$8.6 \cdot 10^{-9}$	$8.5 \cdot 10^{-9}$	$8.6 \cdot 10^{-9}$	$8.1 \cdot 10^{-9}$	$8.8 \cdot 10^{-9}$
Defect	$8.4 \cdot 10^{-8}$	$2.1 \cdot 10^{-7}$	$2.9 \cdot 10^{-7}$	$3.6 \cdot 10^{-8}$	$3.6 \cdot 10^{-8}$	$2.0 \cdot 10^{-8}$	$3.6 \cdot 10^{-8}$

TABLE 5.9
COLNEW on Problem 2*: fourth order,* tol = 10^{-6}.

$\gamma \Rightarrow$	0.0	5.0	10.0	20.0	50.0	100.0
Profile	(5, 7)	(5, 5)	(5, 6)	(5, 10)(320, 1)	(5, 14)(40, 1)	(5, 13)(40, 1)
	(10, 1)	(10, 2)	(10, 2)	(10, 3)	(10, 7)(80, 1)	(10, 8)(40, 1)
	(10, 1)	(10, 1)	(10, 1)	(10, 2)	(10, 2)(160, 1)	(10, 6)(80, 1)
	(20, 1)	(20, 1)	(20, 1)	(20, 1)	(20, 1)(320, 1)	(10, 2)(160, 1)
	(40, 1)	(40, 1)	(40, 1)	(40, 1)	(20, 1)	(20, 1)(320, 1)
		(80, 1)	(80, 1)	(80, 1)	(20, 1)	(20, 1)(160, 1)
		(160, 1)	(160, 1)	(160, 1)	(40, 1)	(20, 1)(320, 1)
Time	0.304	0.988	1.015	1.998	2.360	3.763
Error	$2.4 \cdot 10^{-7}$	$4.7 \cdot 10^{-8}$	$6.0 \cdot 10^{-7}$	$4.8 \cdot 10^{-8}$	$8.1 \cdot 10^{-8}$	$5.2 \cdot 10^{-8}$
Defect	$7.5 \cdot 10^{-5}$	$1.9 \cdot 10^{-4}$	$1.6 \cdot 10^{-3}$	$1.5 \cdot 10^{-3}$	$2.5 \cdot 10^{-3}$	$8.2 \cdot 10^{-3}$

TABLE 5.10
MIRKDC on Problem 2*: fourth order, error* = 10^{-6}.

$\gamma \Rightarrow$	0.0	5.0	10.0	20.0	50.0	100.0
tol $\Rightarrow$	$5 \cdot 10^{-7}$	$1 \cdot 10^{-6}$	$6 \cdot 10^{-7}$	$3 \cdot 10^{-7}$	$6 \cdot 10^{-7}$	$1 \cdot 10^{-7}$
Profile	(5, 5)	(5, 3)	(5, 5)	(5, 6)(282, 1)	(5, *)(282, 1)	(5, *)(459, 1)
	(20, 1)	(20, 1)	(20, 2)	(20, 2)	(10, 6)(313, 1)	(10, 7)(554, 1)
	(36, 1)	(80, 1)	(80, 1)	(80, 1)	(40, 2)(334, 1)	(40, 3)(609, 1)
		(111, 1)	(163, 1)	(234, 1)	(160, 1)	(160, 1)
Time	0.151	0.730	0.623	0.884	2.303	2.161
Error	$9.9 \cdot 10^{-7}$	$7.1 \cdot 10^{-7}$	$8.9 \cdot 10^{-7}$	$7.9 \cdot 10^{-7}$	$7.9 \cdot 10^{-7}$	$7.0 \cdot 10^{-7}$
Defect	$6.0 \cdot 10^{-7}$	$9.4 \cdot 10^{-7}$	$1.2 \cdot 10^{-6}$	$3.1 \cdot 10^{-7}$	$5.3 \cdot 10^{-7}$	$1.6 \cdot 10^{-7}$

The third problem family is the Shock Wave family from [Ascher, Mattheij, and Russell, 1988, p. 21]. It includes the problem parameter, ϵ, the inverse of the Reynold's number. For our testing, we chose ϵ = 1.0, 0.1, 0.05, 0.02, 0.0175. The results are given in Tables 5.17-5.24.

TABLE 5.11
COLNEW on Problem 2: *fourth order,* tol $= 10^{-8}$.

$\gamma \Rightarrow$	0.0	5.0	10.0	20.0	50.0	100.0
Profile	(5, 7)	(5, 6)(320, 1)	(5, 7)(320, 1)	(5, 11)(320, 1)	(5, 14)(80, 1)	(5, 13)(40, 1)
	(10, 1)	(10, 2)	(10, 2)(640, 1)	(10, 3)(268, 1)	(10, 7)(80, 1)	(10, 8)(80, 1)
	(10, 1)	(10, 1)	(10, 2)	(10, 2)(536, 1)	(10, 3)(160, 1)	(10, 7)(160, 1)
	(20, 1)	(20, 1)	(20, 1)	(20, 1)	(20, 2)(320, 1)	(10, 2)(320, 1)
	(40, 1)	(40, 1)	(40, 1)	(40, 1)	(20, 1)(320, 1)	(20, *)(320, 1)
	(80, 1)	(80, 1)	(80, 1)	(80, 1)	(20, 1)(640, 1)	(40, 3)(320, 1)
	(160, 1)	(160, 1)	(160, 1)	(160, 1)	(40, 1)	(40, 1)(640, 1)
Time	0.887	1.948	3.706	4.176	5.096	6.423
Error	$9.1 \cdot 10^{-10}$	$3.7 \cdot 10^{-9}$	$2.1 \cdot 10^{-9}$	$1.5 \cdot 10^{-9}$	$2.0 \cdot 10^{-9}$	$3.3 \cdot 10^{-9}$
Defect	$1.3 \cdot 10^{-6}$	$2.4 \cdot 10^{-5}$	$2.5 \cdot 10^{-5}$	$5.3 \cdot 10^{-5}$	$1.9 \cdot 10^{-4}$	$6.7 \cdot 10^{-4}$

TABLE 5.12
MIRKDC on Problem 2: *fourth order,* error $= 10^{-8}$.

$\gamma \Rightarrow$	0.0	5.0	10.0	20.0	50.0	100.0
tol $\Rightarrow$	$4 \cdot 10^{-9}$	$1 \cdot 10^{-8}$	$7 \cdot 10^{-9}$	$3 \cdot 10^{-9}$	$5 \cdot 10^{-9}$	$9 \cdot 10^{-10}$
Profile	(5, 5)	(5, 3)(356, 1)	(5, 5)(516, 1)	(5, 6)(752, 1)	(5, *)(640, 1)	(5, *)(640, 1)
	(20, 1)	(20, 1)	(20, 2)	(20, 2)(888, 1)	(10, 6)(949, 1)	(10, 7)(1508, 1)
	(80, 1)	(80, 1)	(80, 1)	(80, 1)	(40, 2)(1043, 1)	(40, 3)(1783, 1)
	(124, 1)	(278, 1)	(320, 1)	(320, 1)	(160, 1)	(160, 1)(1961, 1)
Time	0.235	1.053	1.044	1.713	3.351	3.233
Error	$9.0 \cdot 10^{-9}$	$6.6 \cdot 10^{-9}$	$9.2 \cdot 10^{-9}$	$8.3 \cdot 10^{-9}$	$9.2 \cdot 10^{-9}$	$6.4 \cdot 10^{-9}$
Defect	$3.6 \cdot 10^{-9}$	$8.0 \cdot 10^{-9}$	$7.9 \cdot 10^{-9}$	$2.5 \cdot 10^{-9}$	$3.9 \cdot 10^{-9}$	$9.0 \cdot 10^{-10}$

TABLE 5.13
COLNEW on Problem 2: *sixth order,* tol $= 10^{-6}$.

$\gamma \Rightarrow$	0.0	5.0	10.0	20.0	50.0	100.0
Profile	(5, 7)	(5, 5)	(5, 7)(28, 1)	(5, 11)(80, 1)	(5, 19)(40, 1)	(5, *)
	(10, 1)	(10, 1)	(10, 1)	(10, 2)	(10, 2)(21, 1)	*FAIL*
	(20, 1)	(20, 1)	(20, 1)	(20, 1)	(10, 1)(42, 1)	
		(40, 1)	(14, 1)	(40, 1)	(20, 1)	
Time	0.433	0.644	0.760	1.524	1.671	–
Error	$7.4 \cdot 10^{-8}$	$1.2 \cdot 10^{-7}$	$1.1 \cdot 10^{-7}$	$6.1 \cdot 10^{-9}$	$1.7 \cdot 10^{-7}$	–
Defect	$9.6 \cdot 10^{-6}$	$2.5 \cdot 10^{-5}$	$4.4 \cdot 10^{-4}$	$4.0 \cdot 10^{-4}$	$3.6 \cdot 10^{-3}$	–

TABLE 5.14
MIRKDC on Problem 2: *sixth order, error* $= 10^{-6}$.

$\gamma \Rightarrow$	0.0	5.0	10.0	20.0	50.0	100.0
tol $\Rightarrow$	$1 \cdot 10^{-5}$	$1 \cdot 10^{-5}$	$3 \cdot 10^{-5}$	$1 \cdot 10^{-5}$	$3 \cdot 10^{-5}$	$5 \cdot 10^{-6}$
Profile	(5, 5)	(5, 9)	(5, *)	(5, *)(61, 1)	(5, *)(63, 1)	(5, *)(89, 1)
	(12, 1)	(20, 13)	(10, 5)	(10, *)(67, 1)	(10, *)(69, 1)	(10, *)(97, 1)
	(14, 1)	(35, 1)	(34, 1)	(20, 5)	(20, *)(75, 1)	(20, *)(106, 1)
		(38, 1)	(41, 1)	(52, 1)	(40, 20)	(40, 12)(116, 1)
Time	0.106	0.624	0.318	0.901	2.013	1.791
Error	$4.7 \cdot 10^{-7}$	$2.0 \cdot 10^{-7}$	$8.5 \cdot 10^{-7}$	$4.0 \cdot 10^{-7}$	$6.5 \cdot 10^{-7}$	$6.0 \cdot 10^{-7}$
Defect	$2.4 \cdot 10^{-6}$	$2.8 \cdot 10^{-6}$	$1.9 \cdot 10^{-5}$	$3.8 \cdot 10^{-6}$	$2.0 \cdot 10^{-5}$	$3.2 \cdot 10^{-6}$

5.3. Discussion of results. These results demonstrate that the observed performance of MIRKDC can be comparable to that of COLNEW over a range of accuracy requests and problem difficulties, when we consider first-order systems with uniform tolerance requirements.

TABLE 5.15
COLNEW on Problem 2: sixth order, tol = 10^{-8}.

$\gamma \Rightarrow$	0.0	5.0	10.0	20.0	50.0	100.0
Profile	(5, 7)(20, 1)	(5, 5)(23, 1)	(5, 7)(40, 1)	(5, 12)(40, 1)	(5, 19)(40, 1)	(5, *)
	(10, 1)	(10, 1)(46, 1)	(10, 1)(80, 1)	(10, 2)(80, 1)	(10, 3)(40, 1)	*FAIL*
	(20, 1)	(20, 1)	(20, 1)	(10, 1)(160, 1)	(10, 2)(80, 1)	
	(10, 1)	(40, 1)	(20, 1)	(20, 1)	(20, 1)	
Time	0.541	1.039	1.354	2.592	2.460	–
Error	$1.0 \cdot 10^{-9}$	$1.5 \cdot 10^{-9}$	$5.5 \cdot 10^{-10}$	$1.1 \cdot 10^{-10}$	$2.6 \cdot 10^{-9}$	–
Defect	$3.3 \cdot 10^{-7}$	$2.3 \cdot 10^{-6}$	$6.0 \cdot 10^{-6}$	$1.8 \cdot 10^{-5}$	$2.0 \cdot 10^{-4}$	–

TABLE 5.16
MIRKDC on Problem 2: sixth order, error = 10^{-8}.

$\gamma \Rightarrow$	0.0	5.0	10.0	20.0	50.0	100.0
tol $\Rightarrow$	$2 \cdot 10^{-7}$	$5 \cdot 10^{-7}$	$5 \cdot 10^{-7}$	$1 \cdot 10^{-7}$	$2 \cdot 10^{-7}$	$5 \cdot 10^{-8}$
Profile	(5, 5)	(5, 9)	(5, *)(82, 1)	(5, *)(125, 1)	(5, *)(121, 1)	(5, *)(160, 1)
	(20, 1)	(20, 13)	(10, 5)	(10, *)(137, 1)	(10, *)(143, 1)	(10, *)(201, 1)
	(25, 1)	(53, 1)	(40, 1)	(20, 5)	(20, *)(157, 1)	(20, *)(221, 1)
		(61, 1)	(75, 1)	(80, 1)	(40, 20)	(40, 12)(243, 1)
Time	0.152	0.756	0.627	1.224	2.586	2.880
Error	$8.5 \cdot 10^{-9}$	$9.1 \cdot 10^{-9}$	$9.4 \cdot 10^{-9}$	$4.6 \cdot 10^{-9}$	$8.9 \cdot 10^{-9}$	$6.7 \cdot 10^{-9}$
Defect	$7.0 \cdot 10^{-8}$	$1.6 \cdot 10^{-7}$	$1.3 \cdot 10^{-7}$	$2.7 \cdot 10^{-8}$	$9.2 \cdot 10^{-8}$	$3.3 \cdot 10^{-8}$

TABLE 5.17
COLNEW on Problem 3: fourth order, tol = 10^{-6}.

$\epsilon \Rightarrow$	1.0	0.1	0.05	0.02	0.0175
Profile	(5, 4)	(5, 7)	(5, 8)	(5, *)(58, 1)	(5, 13)
	(10, 1)	(10, 1)	(10, 2)	(10, *)(116, 1)	*FAIL*
		(20, 1)	(20, 1)	(20, 13)	
		(20, 1)	(20, 1)	(20, 2)	
		(40, 1)	(40, 1)	(40, 1)	
		(80, 1)	(80, 1)	(80, 1)	
Time(sec)	0.023	0.170	0.186	0.537	–
Error	$7.0 \cdot 10^{-7}$	$1.2 \cdot 10^{-7}$	$5.1 \cdot 10^{-7}$	$2.7 \cdot 10^{-7}$	–
Defect	$1.3 \cdot 10^{-4}$	$2.5 \cdot 10^{-4}$	$2.5 \cdot 10^{-3}$	$2.4 \cdot 10^{-3}$	–

TABLE 5.18
MIRKDC on Problem 3: fourth order, error = 10^{-6}.

$\epsilon \Rightarrow$	1.0	0.1	0.05	0.02	0.0175
tol $\Rightarrow$	$1 \cdot 10^{-5}$	$6 \cdot 10^{-6}$	$1 \cdot 10^{-6}$	$8 \cdot 10^{-7}$	$8 \cdot 10^{-7}$
Profile	(5, 1)	(5, 3)	(5, *)	(5, *)(214, 1)	(5, *)(242, 1)
	(11, 1)	(20, 13)	(10, 7)	(10, *)(235, 1)	(10, *)
		(45, 1)	(40, 1)	(20, *)	(20, *)
		(53, 1)	(105, 1)	(40, *)	(40, 21)
			(128, 1)	(80, 18)	(160, 1)
			(140, 1)	(173, 1)	(220, 1)
Time(sec)	0.013	0.158	0.228	1.074	0.735
Error	$7.5 \cdot 10^{-7}$	$9.7 \cdot 10^{-7}$	$3.7 \cdot 10^{-7}$	$8.9 \cdot 10^{-7}$	$1.0 \cdot 10^{-6}$
Defect	$5.8 \cdot 10^{-6}$	$3.9 \cdot 10^{-6}$	$4.9 \cdot 10^{-7}$	$4.1 \cdot 10^{-7}$	$4.5 \cdot 10^{-7}$

The mesh selection strategy and stopping criterion of COLNEW are conservative and force an iteration on a final mesh of N subintervals (where N is the number of subintervals reported in the performance profiles). This final iteration, required primarily to obtain a reliable error

TABLE 5.19
COLNEW on Problem 3: *fourth order,* tol $= 10^{-8}$.

$\epsilon \Rightarrow$	1.0	0.1	0.05	0.02	0.0175
Profile	(5, 4) (10, 1) (20, 1) (40, 1)	(5, 7) (10, 1) (20, 1) (20, 1) (40, 1) (80, 1) (160, 1)	(5, 9)(320, 1) (10, 2) (20, 1) (20, 1) (40, 1) (80, 1) (160, 1)	(5, *)(160, 1) (10, *)(320, 1) (20, 13) (20, 2) (40, 1) (80, 1) (80, 1)	(5, 15) *FAIL*
Time(sec)	0.069	0.301	0.571	0.865	–
Error	$2.1 \cdot 10^{-9}$	$8.0 \cdot 10^{-9}$	$6.8 \cdot 10^{-10}$	$5.2 \cdot 10^{-9}$	–
Defect	$2.1 \cdot 10^{-6}$	$3.0 \cdot 10^{-5}$	$3.4 \cdot 10^{-5}$	$2.3 \cdot 10^{-4}$	–

TABLE 5.20
MIRKDC on Problem 3: *fourth order, error* $= 10^{-8}$.

$\epsilon \Rightarrow$	1.0	0.1	0.05	0.02	0.0175
tol $\Rightarrow$	$1 \cdot 10^{-7}$	$3 \cdot 10^{-8}$	$1 \cdot 10^{-8}$	$6 \cdot 10^{-9}$	$5 \cdot 10^{-9}$
Profile	(5, 1) (20, 1) (34, 1)	(5, 3) (20, 1) (80, 1) (171, 1) (199, 1)	(5, *) (10, 7) (40, 1) (160, 1) (348, 1) (407, 1)	(5, *) (10, *) (20, *) (40, *) (80, 18) (320, 1) (643, 1) (742, 1)	(5, *) (10, *) (20, *) (40, 21) (160, 1) (606, 1) (788, 1) (866, 1)
Time(sec)	0.034	0.289	0.599	1.754	1.281
Error	$7.2 \cdot 10^{-9}$	$5.3 \cdot 10^{-9}$	$5.3 \cdot 10^{-9}$	$9.1 \cdot 10^{-9}$	$7.1 \cdot 10^{-9}$
Defect	$7.4 \cdot 10^{-8}$	$2.2 \cdot 10^{-8}$	$7.3 \cdot 10^{-9}$	$4.3 \cdot 10^{-9}$	$2.8 \cdot 10^{-9}$

TABLE 5.21
COLNEW on Problem 3: *sixth order,* tol $= 10^{-6}$.

$\epsilon \Rightarrow$	1.0	0.1	0.05	0.02	0.0175
Profile	(5, 4) (10, 1)	(5, 8) (10, 1) (20, 1)	(5, 10) (10, 1) (20, 1) (10, 1) (20, 1)	(5, *) *FAIL*	(5, 22) (10, *) (20, 8) (14, 1) (28, 1) (56, 1)
Time(sec)	0.041	0.102	0.161	–	0.534
Error	$5.8 \cdot 10^{-10}$	$5.1 \cdot 10^{-7}$	$3.5 \cdot 10^{-7}$	–	$2.0 \cdot 10^{-8}$
Defect	$2.2 \cdot 10^{-7}$	$2.0 \cdot 10^{-4}$	$2.5 \cdot 10^{-4}$	–	$2.1 \cdot 10^{-5}$

estimate (after a convergent solution on $N/2$ subintervals has been determined), can result in a solution that is more accurate than requested at a cost that is higher than necessary. This behavior leads to costs being very sensitive to small changes in the accuracy request (or to any other problem parameter) and therefore makes direct comparison of results very difficult. This must be kept in mind when interpreting our numerical results. The error control strategy and mesh selection strategy of MIRKDC are more flexible than that of COLNEW (allowing a more finely adapted selection of an appropriate mesh) and leads to a generally smoother relationship between the cost and requested accuracy.

TABLE 5.22
MIRKDC on Problem 3: *sixth order, error* = 10^{-6}.

$\epsilon \Rightarrow$ tol $\Rightarrow$	1.0 $3 \cdot 10^{-4}$	0.1 $6 \cdot 10^{-5}$	0.05 $8 \cdot 10^{-5}$	0.02 $4 \cdot 10^{-5}$	0.0175 $2 \cdot 10^{-4}$
Profile	$(5, 1)$	$(5, 4)$ $(15, 1)$ $(19, 1)$	$(5, *)$ $(10, 7)$ $(23, 1)$ $(27, 1)$ $(29, 1)$	$(5, *)$ $(10, *)$ $(20, 21)$ $(43, 2)$ $(51, 1)$	$(5, 13)$ $(20, 5)$ $(40, 22)$ $(44, 1)$ $(48, 1)$ $(52, 1)$
Time(sec)	0.009	0.066	0.181	0.421	0.760
Error	$5.0 \cdot 10^{-7}$	$5.8 \cdot 10^{-7}$	$1.0 \cdot 10^{-6}$	$1.5 \cdot 10^{-6}$	$1.3 \cdot 10^{-6}$
Defect	$9.8 \cdot 10^{-6}$	$2.4 \cdot 10^{-5}$	$3.1 \cdot 10^{-5}$	$2.2 \cdot 10^{-5}$	$3.0 \cdot 10^{-5}$

TABLE 5.23
COLNEW on Problem 3: *sixth order,* tol = 10^{-8}.

$\epsilon \Rightarrow$	1.0	0.1	0.05	0.02	0.0175
Profile	$(5, 4)$ $(10, 1)$	$(5, 8)$ $(10, 1)$ $(20, 1)$ $(13, 1)$ $(26, 1)$	$(5, 11)$ $(10, 2)$ $(20, 1)$ $(20, 1)$ $(40, 1)$	$(5, *)$ *FAIL*	$(5, 22)$ $(10, *)$ $(20, 9)$ $(20, 1)$ $(40, 1)$ $(80, 1)$
Time(sec)	0.037	0.159	0.230	–	0.650
Error	$5.8 \cdot 10^{-10}$	$4.7 \cdot 10^{-9}$	$7.7 \cdot 10^{-9}$	–	$2.7 \cdot 10^{-9}$
Defect	$2.2 \cdot 10^{-7}$	$4.0 \cdot 10^{-6}$	$7.7 \cdot 10^{-6}$	–	$7.2 \cdot 10^{-6}$

TABLE 5.24
MIRKDC on Problem 3: *sixth order, error* = 10^{-8}.

$\epsilon \Rightarrow$ tol $\Rightarrow$	1.0 $2 \cdot 10^{-6}$	0.1 $6 \cdot 10^{-7}$	0.05 $5 \cdot 10^{-7}$	0.02 $3 \cdot 10^{-7}$	0.0175 $3 \cdot 10^{-7}$
Profile	$(5, 1)$ $(8, 1)$	$(5, 4)$ $(20, 1)$ $(37, 1)$	$(5, *)$ $(10, 7)$ $(40, 1)$ $(60, 1)$	$(5, *)$ $(10, *)$ $(20, 21)$ $(80, 2)$ $(111, 1)$	$(5, 13)$ $(20, 5)$ $(40, 22)$ $(103, 1)$ $(121, 1)$ $(133, 1)$
Time(sec)	0.023	0.102	0.228	0.592	1.094
Error	$9.6 \cdot 10^{-9}$	$9.5 \cdot 10^{-9}$	$9.4 \cdot 10^{-9}$	$1.0 \cdot 10^{-8}$	$7.7 \cdot 10^{-9}$
Defect	$4.1 \cdot 10^{-7}$	$3.9 \cdot 10^{-7}$	$6.1 \cdot 10^{-7}$	$4.1 \cdot 10^{-7}$	$1.5 \cdot 10^{-7}$

From the results presented here one can observe that MIRKDC can be as efficient and can exhibit smoother and more predictable behavior as we consider more difficult problems in the same problem family. There are some instances (for both methods) where the cost is unexpectedly large or no results are available. By studying the corresponding performance profiles we see that the former behavior is usually the result of a large number of Newton iterations associated with a coarse-mesh approximate solution, while the latter behavior is the result of a Newton iteration failure at all attempted meshes. In practical applications these two related difficulties are not uncommon, and special method-independent techniques are often used to overcome them. For example, when the Newton iteration fails to converge for any mesh one can obtain more accurate initial guesses by employing continuation with respect to the problem parameter that quantifies the level of difficulty.

Note that the rather large defects observed in the performance profiles associated with COLNEW should not be interpreted as a deficiency. This code was not designed to control this quantity, and a different interpolating strategy would lead to a much improved relationship between the size of the defect and the requested accuracy. This statistic is only reported to confirm that MIRKDC is controlling the defect in the way that is intended.

6. Conclusions and future work. In this paper we have discussed an alternative approach for the numerical solution of BVODE problems. The major new aspects are the use of continuous MIRK formulas and the use of defect estimation rather than global error estimation for accuracy control and mesh selection. The algorithms for this alternate approach have been implemented in a code called MIRKDC which, in some preliminary testing, has demonstrated performance comparable to that of the widely used software package, COLNEW. The results are sufficiently encouraging to indicate that further work in the development of MIRKDC is worthwhile. Further testing on a larger set of test problems is now proceeding.

There are several aspects of the new approach and software that require further investigation. Foremost among these is the need to explore the space of available MIRK formulas to select specific optimal formulas of each order. A related question involves a theoretical analysis of CMIRK schemes containing optimal embedded MIRK schemes. It is clear that the order barriers in such a case may be different than those of [Muir and Owren, 1993], where the restriction of an embedded optimal discrete formula is not included. The new nine-stage, sixth-order CMIRK scheme presented in §3 is an example of this kind of scheme. Subsequent to the characterization of optimal CMIRK formulas with optimal embedded discrete MIRK formulas, a study to explore the parameter space of the CMIRK families to discover appropriate specific formulas is needed.

Additional areas for future work include further investigation of the parameters which control the Newton iteration since the overall efficiency and robustness of a BVODE code is very much dependent on the performance of this algorithm. Further investigation of controlling parameters within the mesh redistribution algorithm and in the algorithm controlling continuation of solutions and meshes over the discrete problem sequences may also lead to significant performance improvements. A number of questions remain to be answered concerning the use of defect control in the numerical solution of BVODE problems. A variety of defect estimation schemes are available and the most appropriate choice for the BVODE context remains to be established. Also, the use of defect control allows more freedom in mesh selection, which in turn appears to lead to some performance advantages which must be further explored.

REFERENCES

U. Ascher, J. Christiansen, and R. D. Russell, *Collocation software for boundary value ODEs*, Trans. Math. Software, 7 (1981), pp. 209-222.

U. Ascher, R. M. M. Mattheij, and R. Russell, *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*, Prentice Hall, Englewood Cliffs, New Jersey, 1988.

U. Ascher and S. Jacobs, *On collocation implementation for singularly perturbed two-point problems*, SIAM J. Sci. Stat. Comp., 10 (1989), pp. 533-549.

G. Bader and U. Ascher, *A new basis implementation for a mixed order boundary value ode solver*, SIAM J. Sci. Stat. Comp., 8 (1987), pp. 483-500.

G. K. Batchelor, *A note on a class of solutions of the Navier-Stokes equations representing rotationally symmetric flow*, Quart. J. Mech. Appl. Math., 4 (1951), pp. 29-41.

W. M. G. van Bokhoven, *Efficient higher order implicit one-step methods for integration of stiff differential equations*, BIT, 20 (1980), pp. 34-43.

D. L. Brown and J. Lorenz, *A high-order method for stiff boundary value problems with turning points*, SIAM J. Sci. Stat. Comp., 8 (1987), pp. 790-808.

K. Burrage, F. Chipman, and P. H. Muir, *Order results for mono-implicit Runge-Kutta methods*, SIAM J. Numer. Anal., 31 (1994), pp. 876-891.

J. C. Butcher, *The Numerical Analysis of Ordinary Differential Equations*, John Wiley & Sons, Toronto, 1987.

J. R. Cash, *On the numerical integration of nonlinear two-point boundary value problems using iterated deferred corrections, Part* 1: *A survey and comparison of some one-step formulae*, Comp. Math. Appl., 12a (1986), pp. 1029-1048.

———, *On the numerical integration of nonlinear two-point boundary value problems using iterated deferred corrections, Part* 2: *The development and analysis of highly stable deferred correction formulae*, SIAM J. Numer. Anal., 25 (1988), pp. 862-882.

J. R. Cash and H. H. M. Silva, *On the numerical solution of a class of singular two-point boundary value problems*, J. Comput. Appl.Math., 45 (1993), pp. 91-102.

J. R. Cash and A. Singhal, *Mono-implicit Runge-Kutta formulae for the numerical integration of stiff differential systems*, IMA J. Numer. Anal., 2 (1982a), pp. 211-227.

———, *High order methods for the numerical solution of two-point boundary value problems*, BIT, 22 (1982b), pp. 184-199.

J. R. Cash and M. H. Wright, *Implementation issues in solving nonlinear equations for two-point boundary value problems*, Computing, 45 (1990), pp. 17-37.

———, *A deferred correction method for nonlinear two-point boundary value problems: implementation and numerical evaluation*, SIAM J. Sci. Stat. Comp., 12 (1991), pp. 971-989.

W. H. Enright, *A new error-control for initial value solvers*, Appl. Math. Comp., 31 (1989a), pp. 288-301.

———, *Analysis of error control strategies for continuous Runge-Kutta methods*, SIAM J. Numer. Anal., 26 (1989b), pp. 588-599.

W. H. Enright, K. R. Jackson, S. P. Nørsett, and P. G. Thomsen, *Interpolants for Runge-Kutta formulas*, ACM Trans. Math. Software, 12 (1986), pp. 193-218.

W. H. Enright and P. H. Muir, *Efficient classes of Runge-Kutta methods for two-point boundary value problems*, Computing, 37 (1986), pp. 315-334.

———, *A Runge-Kutta type boundary value ODE solver with defect control*, Tech. Rep. No. 267/93, Dept. Comp. Sci., Univ. of Toronto, Toronto, Ontario, Canada, 1993.

I. Gladwell, L. F. Shampine, L. S. Baca, and R. W. Brankin, *Practical aspects of interpolation in Runge-Kutta codes*, SIAM J. Sci. Stat. Comp., 8 (1987), pp. 322-341.

S. Gupta, *An adaptive boundary value Runge-Kutta solver for first order boundary value problems*, SIAM J. Numer. Anal., 22 (1985), pp. 114-126.

P. M. Hanson and W. H. Enright, *Controlling the defect in existing variable-order Adams codes for initial-value problems*, ACM Trans. Math. Software, 9 (1983), pp. 71-97.

D. J. Higham, *Defect estimation in Adams PECE codes*, SIAM J. Sci. Stat. Comput., 10 (1989a), pp. 964-976.

———, *Robust defect control with Runge-Kutta schemes*, SIAM J. Numer. Anal., 26 (1989b), pp. 1175-1183.

———, *Runge-Kutta defect control using Hermite-Birkhoff interpolation*, SIAM J. Sci. Stat. Comput., 12 (1991), pp. 991-999.

H.-O. Kreiss, N. K. Nichols, and D. L. Brown, *Numerical methods for stiff two-point boundary value problems*, SIAM J. Numer. Anal., 23 (1986), pp. 325-368.

M. Lentini and V. Pereyra, *An adaptive finite difference solver for nonlinear two-point boundary value problems with mild boundary layers*, SIAM J. Numer. Anal., 14 (1977), pp. 91-111,

P. H. Muir and B. Owren, *Order barriers and characterizations of continuous mono-implicit Runge-Kutta schemes*, Math. Comp., 61 (1993), pp. 675-699.

B. Owren and M. Zennaro, *Order barriers for continuous explicit Runge-Kutta methods*, Math. Comp., 56 (1991), pp. 645-661.

———, *Derivation of optimal continuous explicit Runge-Kutta methods*, SIAM J. Sci. Stat. Comp., 13 (1992), pp. 1488-1501.

V. Pereyra and R. D. Russell, *Difficulties of comparing complex mathematical software: General comments and the BVODE case*, Acta Cient. Venezolana, 33 (1982), pp. 15-22.

A. Prothero and A. Robinson, *On the stability and accuracy of one-step methods for solving stiff systems of ordinary differential equations*, Math. Comp., 28 (1974), pp. 145-162.

P. W. Sharp and E. Smart, *Explicit Runge-Kutta pairs with one more derivative evaluation than the minimum*, SIAM J. Sci. Comp., 14 (1993), pp. 338-348.

J. Verner, *Differentiable interpolants for high-order Runge-Kutta methods*, SIAM J. Numer. Anal., 30 (1993), pp. 1446-1466.

R. Weiss, *The application of implicit Runge-Kutta and collocation methods to boundary value problems*, Math. Comp., 28 (1974), pp. 449-464.

SIAM J. SCI. COMPUT.
Vol. 17, No. 2, pp. 498-516, March 1996

012

THE DIFFERENTIATION MATRIX FOR DAUBECHIES-BASED WAVELETS ON AN INTERVAL*

LELAND JAMESON†

Abstract. The differentiation matrix for a Daubechies-based wavelet basis defined on an interval is constructed. It is shown that the differentiation matrix based on the currently available boundary constructions does not maintain the superconvergence encountered under periodic boundary conditions.

Key words. wavelet approximation, differentiation matrix, superconvergence

AMS subject classifications. 65N30, 65D25

1. Introduction. Superconvergence is a property of wavelet methods, which is Daubechies- (see [4]), and spline-based (see [10]), when periodic boundary conditions are assumed and a Galerkin projection is used. That is, for Daubechies-based methods under the assumption of periodicity, it was proven in [6] that the differentiation matrix is accurate of order $2M$, even though the highest degree polynomial that can be reconstructed exactly in the wavelet subspace is of degree $M - 1$, where M is the number of vanishing moments of the wavelet. For spline-based methods, again assuming periodicity, it was proven in [7] that the differentiation matrix is accurate of order $2n + 2$, even though one can only construct splines of order n exactly in the underlying subspace. For Daubechies- and spline-based wavelet systems this approximate doubling of the differentiation accuracy is known as superconvergence, a name borrowed from the finite element literature.

When periodicity is no longer assumed and one defines wavelets on an interval, one of the goals when building boundary functions should be to maintain the superconvergence of the differentiation matrix across the entire interval. It has been proven by Gottlieb et al. [5] that the superconvergence encountered in finite element methods under the assumption of periodicity cannot be maintained on the interval if there are characteristics leaving the domain. Furthermore, it was shown in [7] that for spline-based wavelet systems superconvergence is lost at the boundaries when the boundary functions are truncated B-splines. The goal of this paper is to explore the accuracy of the differentiation matrix for Daubechies-based wavelet systems defined on an interval. In particular, I will explore the differentiation accuracy of the differentiation matrices constructed from the currently available boundary constructions for Daubechies wavelets defined on an interval. The differentiation matrix for the Daubechies-4 (D_4)wavelet basis using the boundary functions defined in [3] will be calculated explicitly and the accuracy will be found. From this explicit example one can see a necessary condition for superconvergence to be maintained with other boundary constructions.

We begin by defining the differentiation matrix $\mathcal{D}$ as the product of three matrices: $\mathcal{D} = \hat{C}DC$. The three matrices are defined as follows:

- The first matrix C is a quadrature matrix mapping from samples in the physical space to approximate scaling function coefficients at the finest scale: $\vec{s} = C\vec{f}$. C can be derived from the moments of the scaling function. C, also, can be simply the inverse of the third matrix: $C = (\hat{C})^{-1}$.

*Received by the editors December 15, 1993; accepted for publication (in revised form) November 7, 1994.

†Mitsubishi Heavy Industries, Ltd., Advanced Technology Research Center, 8-1, Sachiura, 1-Chome, Kanazawa-ku, Yokohama, 236, Japan (lmj@atrc.ykhm.mhi.co.ip). This research was supported by and conducted at the National Aeronautics and Space Administration under NASA Contract NAS1-19480 while the author was in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA 23681. Research was also supported by AFOSR grant 93-1-0090, by DARPA grant N00014-91-4016, and by NSF grant DMS-9211820.

- The second matrix D maps from the scaling function coefficients of a function to the scaling function coefficients of the derivative of the same function. For convenience I will, henceforth, refer to this matrix as the "derivative projection" matrix.
- The third matrix $\hat{C}$ is a quadrature matrix mapping from approximate scaling function coefficients to approximate point values in the physical space: $\vec{f} = \hat{C}\vec{s}$. $\hat{C}$ can be derived from samples of the scaling function or can simply be the inverse of the first matrix: $\hat{C} = C^{-1}$.

This paper contains the following sections.

1. **Introduction**
2. **Boundary functions.** An outline of the boundary functions constructed in [3] is given.
3. **Derivative projection matrix D.** The entries of D are defined and calculated. D has a banded structure due to the local support of the basis functions, but D is not antisymmetric. The number of nonzero entries in each row of D bounds the maximum differentiation accuracy for the corresponding row in the differentiation matrix. Furthermore, the number of nonzero entries in the first few and last few rows of D are related to the overlap of the boundary functions.
4. **Approximating scaling function coefficients.** An outline of scaling function coefficient approximation is given.
5. **Quadrature matrix C using moments.** C performs the mapping $C : \vec{f} \rightarrow \vec{s}$. The D_4 wavelet has two vanishing moments which yield well-conditioned two-point quadrature matrices and ill-conditioned three-point quadrature matrices.
6. **Quadrature matrix $\hat{C}$ using samples.** $\hat{C}$ performs the mapping, $\hat{C} : \vec{s} \rightarrow \vec{f}$. For the D_4 wavelet the matrix $\hat{C}$ has three entries in each row and is banded. $\hat{C}$, however, is ill conditioned.
7. **Examples and order of accuracy.** The matrix D is fixed. However, there are many choices for the quadrature matrix producing many versions of $\mathcal{D}$. In all cases superconvergence is lost at the boundaries. Furthermore, choosing among the quadrature matrices which are not ill conditioned, one gets a differentiation matrix which is essentially full. A banded $\mathcal{D}$ can be constructed from a combination of the two types of quadrature matrices C and $\hat{C}$. The bandwidth from such a construction is large, 9, for a D_4 wavelet and, again, the superconvergence is lost at the boundaries. In addition, all of these matrices display directional dependency. This directional dependency can be "fixed" by reversing the coordinate system and averaging the appropriate differentiation matrices.
8. **Conclusion.** The differentiation matrices for wavelets on an interval using the boundary functions defined in [3] do not maintain the superconvergence encountered when periodic boundary conditions are assumed. By simply counting the overlap of boundary functions in other boundary constructions it can be seen that all currently available boundary constructions fail to maintain the superconvergence encountered under periodic boundary conditions.

2. Construction of boundary functions. This section is a brief outline of the boundary function construction proposed by Cohen, Daubechies, and Vial; see [3].

The goal is to build a wavelet basis on an interval where the scaling functions and wavelets away from the boundaries are the usual Daubechies scaling functions and wavelets. At the boundaries, boundary scaling functions are constructed such that polynomials of order up to the number of vanishing moments of the wavelet can be reproduced exactly across the entire interval.

The boundary function construction begins by building independent, but not orthogonal, functions as follows:

$$\hat{\phi}^k(x) = \sum_{n=k}^{2N-2} \binom{n}{k} \phi(x+n-N+1), \tag{1}$$

where $\phi(x)$ is the usual Daubechies scaling function and N is the number of vanishing moments of the associated wavelet. Note that these functions are compactly supported, and their supports are staggered, i.e.,

$$\operatorname{supp}(\hat{\phi}^k) = [0, 2N-1-k]. \tag{2}$$

The staggered support yields independence, and the boundary functions are then defined by simply orthonormalizing these functions by a Gram-Schmidt method.

Coefficients h^L and h^R can be found, see [3], such that the boundary functions are defined recursively as follows beginning with the left-hand side boundary functions:

$$\phi_k^L(x) = \sqrt{2}\sum_{l=0}^{N-1} h_{k,l}^L \phi_l^L(2x) + \sqrt{2}\sum_{m=N}^{N+2k} h_{k,m}^L \phi(2x-m), \tag{3}$$

and the equation defining the right-hand side boundary functions is

$$\phi_k^R(x) = \sqrt{2}\sum_{l=0}^{N-1} h_{k,l}^R \phi_l^R(2x) + \sqrt{2}\sum_{m=N}^{N+2k} h_{k,m}^R \phi(2x+m+1). \tag{4}$$

Figure 1 is a plot of the four boundary scaling functions for the D_4 wavelet.

3. The derivative projection matrix D. Let us recall the origin of the matrix D. First, we begin with a function $f(x) \in L^2(R)$. Next, we want to approximate, or discretize, this function in a scaling function subspace $V_0 \subset L^2(R)$,

$$P_{V_0} f(x) = \sum_{k=0}^{d-1} s_k b_k(x), \tag{5}$$

where $\{b_k(x)\}$ denotes the basis functions of this finest scale discretization, and d is the dimension or number of degrees of freedom of the subspace. In the case of wavelets with periodic boundary conditions $b_k(x) = \phi_k(x)$, the usual Daubechies scaling functions, for all $k \in \{0, \dots, d-1\}$. However, in the current scenario of wavelets on an interval, $b_k(x)$ are different at the boundaries:

$$b_k(x) = \phi_k^L(x)$$

for $k = 0, \dots, N-1$,

$$b_k(x) = \phi_k(x)$$

for $N \le k \le d-2N+1$, and

$$b_k(x) = \phi_k^R(x)$$

for $k = d-2N+2, \dots, d-1$, where N is the number of vanishing moments of the wavelet. Under the orthonormality of the current basis set $\langle b_k, b_l\rangle = \delta_{k,l}$, where

$$\langle g, h\rangle \equiv \int_I g(x)h(x)dx, \tag{6}$$

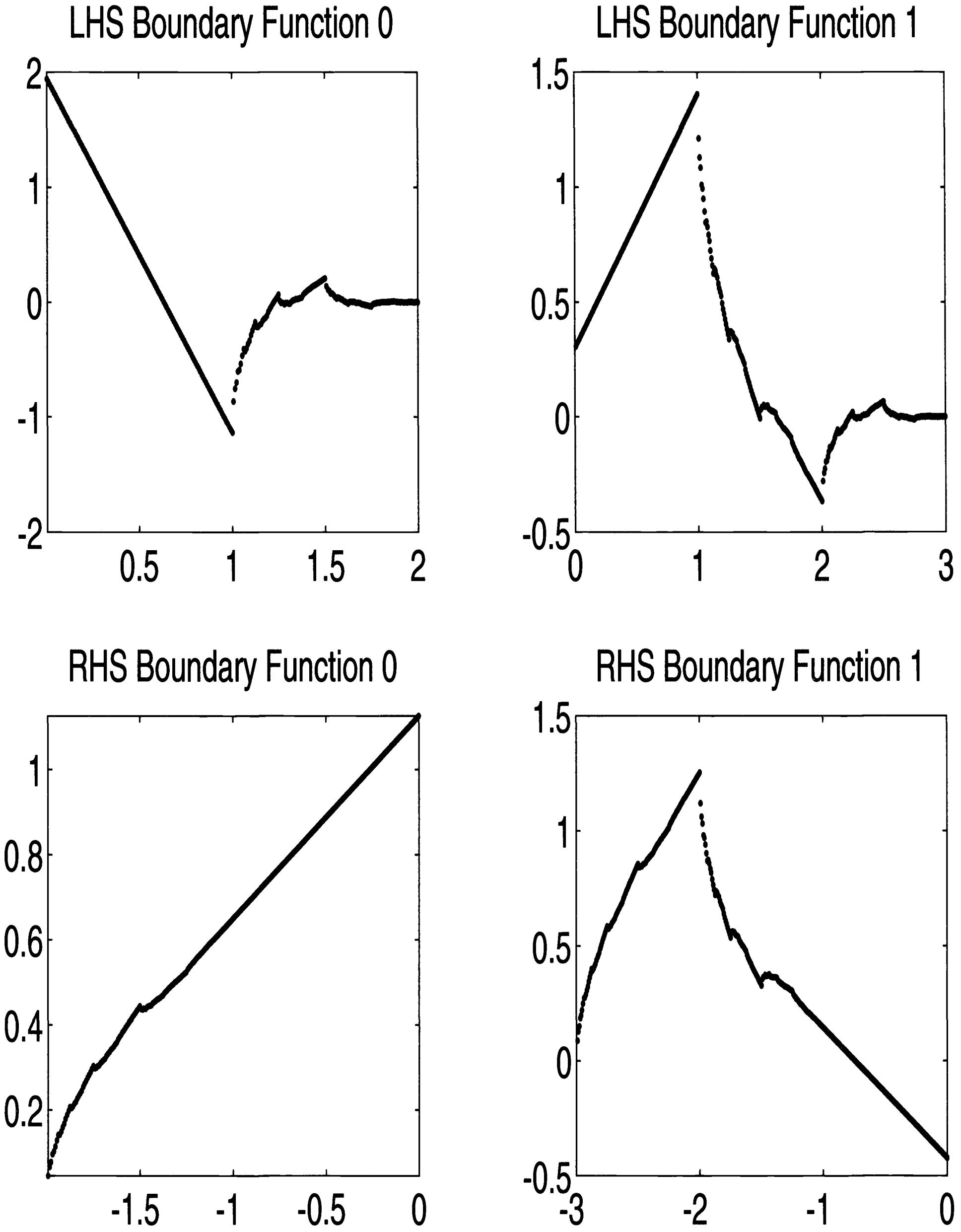


FIG. 1. *The boundary functions for the D_4 scaling function.*

and $\int_I$ denotes integration over the interval, we get

$$s_k = \langle b_k, f \rangle . \tag{7}$$

Now we differentiate the approximation $P_{V_0} f(x)$ to get

$$\frac{d}{dx} P_{V_0} f(x) = \sum_{k=0}^{d-1} s_k \acute{b}_k(x), \tag{8}$$

where $\acute{b}(x) = \frac{d}{dx}b(x)$. The derivative takes $P_{V_0}f(x)$ out of V_0. Projecting back into V_0 we get,

$$P_{V_0}\frac{d}{dx}P_{V_0}f(x) = \sum_{l=0}^{d-1}\left\langle \frac{d}{dx}P_{V_0}f, b_l \right\rangle b_l(x) \tag{9}$$

or

$$P_{V_0}\frac{d}{dx}P_{V_0}f(x) = \sum_{l=0}^{d-1}\sum_{k=0}^{d-1} s_k \left\langle \acute{b}_k, b_l \right\rangle b_l(x). \tag{10}$$

The elements $\langle \acute{b}_k, b_l\rangle$ comprise the derivative projection matrix D. If we let $\vec{s}$ denote the vector of the scaling function coefficients s_k, for $k = 0, \dots, d-1$, then D maps from the scaling function coefficients of a function at the finest scale to the scaling function coefficients of the derivative of the same function:

$$D : \vec{s} \to \vec{\acute{s}}.$$

D, in effect, contains all the derivative information in a wavelet expansion. The elements of D contain the numerical values of the derivative of each scaling function projected onto all scaling functions in the basis. Due to the compact support of the basis functions, the matrix D has a band diagonal form. The following matrix illustrates the form of D for the D_4 wavelet basis with boundary functions defined as in the previous section:

$$D = \begin{bmatrix}
\rho^L_{0,0} & \rho^L_{1,0} & \alpha^L_{2,0} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\rho^L_{0,1} & \rho^L_{1,1} & \alpha^L_{2,1} & \alpha^L_{3,1} & 0 & 0 & 0 & 0 & 0 & 0 \\
\beta^L_{0,2} & \beta^L_{1,2} & r_0 & r_1 & r_2 & 0 & 0 & 0 & 0 & 0 \\
0 & \beta^L_{1,3} & r_{-1} & r_0 & r_1 & r_2 & 0 & 0 & 0 & 0 \\
0 & 0 & r_{-2} & r_{-1} & r_0 & r_1 & r_2 & 0 & 0 & 0 \\
0 & 0 & 0 & r_{-2} & r_{-1} & r_0 & r_1 & r_2 & 0 & 0 \\
0 & 0 & 0 & 0 & r_{-2} & r_{-1} & r_0 & r_1 & \beta^R_{1,4} & 0 \\
0 & 0 & 0 & 0 & 0 & r_{-2} & r_{-1} & r_0 & \beta^R_{1,5} & \beta^R_{0,5} \\
0 & 0 & 0 & 0 & 0 & 0 & \alpha^R_{4,1} & \alpha^R_{5,1} & \rho^R_{1,1} & \rho^R_{0,1} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha^R_{5,0} & \rho^R_{1,0} & \rho^R_{0,0}
\end{bmatrix},$$

where the coefficients r are the derivative projections due to scaling function interaction, the coefficients ρ are the projections of the derivatives of boundary functions projected onto boundary functions, and the coefficients α and β contain the interactions between derivatives of boundary functions and scaling functions.

In the following subsections the entries of D will be found.

3.1. The equations for the derivative projections. To find the elements of D one needs to find the interaction of the derivative of each basis function with every other basis function. Due to the local support of the basis functions, the left-hand side (LHS) boundary functions and the right-hand boundary functions do not interact. (Assuming, of course, that one does not choose an unreasonably small domain relative to the support of the scaling functions.)

Each entry in the matrix D must be found separately, but the calculations are straightforward. A sampling of the relevant derivations will be given below.

3.1.1. Derivation of the LHS equations. Recall that the LHS boundary functions $\phi_k^L(x)$ are generated by

$$\phi_k^L(x) = \sqrt{2}\sum_{l=0}^{N-1} h_{k,l}^L \phi_l^L(2x) + \sqrt{2}\sum_{m=N}^{N+2k} h_{k,m}^L \phi(2x-m). \tag{11}$$

Differentiate the above equation to get

$$\frac{d}{dx}\phi_k^L(x) = 2\sqrt{2}\sum_{l=0}^{N-1} h_{k,l}^L \acute{\phi}_l^L(2x) + 2\sqrt{2}\sum_{m=N}^{N+2k} h_{k,m}^L \acute{\phi}(2x-m). \tag{12}$$

Multiply $\acute{\phi}_k^L(x)$ by $\phi_p^L(x)$ and integrate to get

$$\begin{aligned} \int_I \acute{\phi}_k^L(x)\phi_p^L(x)dx &= 4\sum_{l=0}^{N-1}\sum_{i=0}^{N-1} h_{k,l}^L h_{p,i}^L \int_I \acute{\phi}_l^L(2x)\phi_i^L(2x)dx \\ &+ 4\sum_{m=N}^{N+2k}\sum_{i=0}^{N-1} h_{p,i}^L h_{k,m}^L \int_I \phi_i^L(2x)\acute{\phi}(2x-m)dx \\ &+ 4\sum_{l=0}^{N-1}\sum_{q=N}^{N+2p} h_{k,l}^L h_{p,q}^L \int_I \acute{\phi}_l^L(2x)\phi(2x-q)dx \\ &+ 4\sum_{m=N}^{N+2k}\sum_{q=N}^{N+2p} h_{k,m}^L h_{p,q}^L \int_I \acute{\phi}(2x-m)\phi(2x-q)dx. \end{aligned} \tag{13}$$

Rename the integrals as

$$\rho_{k,p}^L = \int_I \acute{\phi}_k^L(x)\phi_p^L(x)dx, \tag{14}$$

$$\alpha_{m,i}^L = \int_I \acute{\phi}(x-m)\phi_i^L(x)dx, \tag{15}$$

$$\beta_{l,q}^L = \int_I \acute{\phi}_l^L(x)\phi(x-q)dx, \tag{16}$$

$$r_{m,q} = \int_I \acute{\phi}(x-m)\phi(x-q)dx, \tag{17}$$

to get the following system of equations:

$$\begin{aligned} \frac{1}{2}\rho_{k,p}^L &= \sum_{l}^{N-1}\sum_{i}^{N-1} h_{k,l}^L h_{p,i}^L \rho_{l,i}^L + \sum_{i=0}^{N-1}\sum_{m=N}^{N+2k} h_{p,i}^L h_{k,m}^L \alpha_{m,i}^L \\ &+ \sum_{l=0}^{N-1}\sum_{q=N}^{N+2p} h_{k,l}^L h_{p,q}^L \beta_{l,q}^L + \sum_{m=N}^{N+2k}\sum_{q=N}^{N+2p} h_{k,m}^L h_{p,q}^L r_{m,q}. \end{aligned} \tag{18}$$

The numerical values of the coefficients $\{r\}$ were found in [2], and the numerical values of the coefficients ρ^L, α^L, and β^L will be found in the following subsections.

3.2. The coefficients $\rho_{i,i}^L$ for $i = 0, 1$. In this subsection we will find the corner diagonal elements of D. These elements represent the projection of the derivative of each boundary function onto itself.

3.2.1. The coefficients $\rho^L_{0,0}$ and $\rho^L_{1,1}$. For simplicity, allow the lower limit to be 0 and the upper limit be ∞ in the definition of $\rho^L_{k,k}$:

$$\rho^L_{k,k} = \int_0^\infty \phi'^L_k(x)\phi^L_k(x)dx.$$

Apply integration by parts to this equation to get

$$\rho^L_{k,k} = (\phi^L_k(x))^2|_0^\infty - \int_0^\infty \phi^L_k(x)\phi'^L_k(x)dx \tag{19}$$

or

$$2\rho^L_{k,k} = (\phi^L_k(x))^2|_0^\infty. \tag{20}$$

$\phi^L_k(x)$ is 0 at $x = \infty$ leaving only the value at the lower limit

$$\rho^L_{k,k} = -\frac{(\phi^L_k(0))^2}{2}. \tag{21}$$

That is, $\rho^L_{k,k}$ is determined by the value of the boundary function at the boundary. Before evaluating the boundary functions at the boundary it will be shown that $\rho^L_{0,0}$ and $\rho^L_{1,1}$ are related by a constant multiple.

3.2.2. The relationship between $\rho^L_{0,0}$ and $\rho^L_{1,1}$. In (11) set $x = 0$ to get

$$\phi^L_k(0) = \sqrt{2}\sum_{l=0}^{1} h^L_{k,l}\phi^L_l(0) + \sqrt{2}\sum_{m=2}^{2+2k} h^L_{k,m}\phi(-m). \tag{22}$$

For $k = 0$ we get

$$\phi^L_0(0) = \sqrt{2}(h^L_{0,0}\phi^L_0(0) + h^L_{0,1}\phi^L_1(0)), \tag{23}$$

which yields

$$\phi^L_1(0) = c^L\phi^L_0(0), \tag{24}$$

where

$$c^L = \frac{1-\sqrt{2}h^L_{0,0}}{\sqrt{2}h^L_{0,1}}. \tag{25}$$

Recall from above that the $\rho^L_{k,k}$'s can be found from the boundary functions evaluated at the boundary. Combine this information with (24) to get

$$\rho^L_{1,1} = (c^L)^2\rho^L_{0,0}. \tag{26}$$

That is, we only need to evaluate one of the boundary functions at the boundary in order to know $\rho^L_{0,0}$ and $\rho^L_{1,1}$. The following subsection will evaluate the boundary function $\phi^L_0(x)$ at $x = 0$.

3.2.3. Evaluation of the boundary function $\phi_0^L(x)$ at $x = 0$. In this subsection the straightforward evaluation of the boundary function $\phi_0^L(x)$ at $x = 0$ will be given.

In (11) let $k = 1$ and $x = 2$ to get

$$\phi_1^L(2) = \sqrt{2}(h_{1,3}^L\phi(1) + h_{1,4}^L\phi(0)). \tag{27}$$

The values of the scaling function $\phi(x)$ in the above integers are $\phi(0) = 1/2(1 + \sqrt{3})$ and $\phi(1) = 1/2(1 - \sqrt{3})$; see [9]. Using these numbers and the coefficients provided in [3] we get $\phi_1^L(2) = -.3654971039$. Similar to above, let $k = 0$ and $x = 1$ in (11) to get

$$\phi_0^L(1) = \sqrt{2}h_{0,1}^L\phi_1^L(2) + \sqrt{2}h_{0,2}^L\phi(0), \tag{28}$$

and evaluate to get $\phi_0^L(1) = -1.1265992786$. In this way, one can get the following values for the boundary functions: $\phi_1^L(1) = 1.4058555636$, $\phi_0^L(1/2) = .4123639530$. Now we have two values for $\phi_0^L(x)$ at $x = 1/2$ and $x = 1$. Furthermore, the two boundary functions are lines between $x = 0$ and $x = 1$. We can, therefore, find the value of $\phi_0^L(x)$ at $x = 0$: $\phi_0^L(0) = 1.9513271847$. As outlined in the previous subsection we now have sufficient information to find $\rho_{0,0}^L$ and $\rho_{1,1}^L$:

$$\rho_{0,0}^L = -1.9038388908$$

and

$$\rho_{1,1}^L = -.04295212263.$$

The calculation of the remaining elements of the derivative projection matrix is straightforward and will not be given. The next subsection provides the complete matrix D.

3.3. The complete matrix *D*. This section of the paper will be concluded by giving a 10-by-10 version of the matrix D. Only four-digit accuracy will be given for the corner entries ρ, α, and β. The entries r are known exactly.

$$D = \begin{bmatrix}
-1.9038 & .9444 & -.2565 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1.5163 & -.0430 & .6752 & -.0832 & 0 & 0 & 0 & 0 & 0 & 0 \\
.2565 & -.6752 & 0 & \frac{2}{3} & -\frac{1}{12} & 0 & 0 & 0 & 0 & 0 \\
0 & .0832 & -\frac{2}{3} & 0 & \frac{2}{3} & -\frac{1}{12} & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{12} & -\frac{2}{3} & 0 & \frac{2}{3} & -\frac{1}{12} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{12} & -\frac{2}{3} & 0 & \frac{2}{3} & -\frac{1}{12} & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{12} & -\frac{2}{3} & 0 & \frac{2}{3} & -.0765 & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1}{12} & -\frac{2}{3} & 0 & .5825 & -.0397 \\
0 & 0 & 0 & 0 & 0 & 0 & .0765 & -.5825 & .0899 & .3150 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .0397 & -.7936 & .6369
\end{bmatrix}.$$

Note that, as expected, away from the boundaries D is the same as the derivative projection matrix which was found for periodic boundary conditions; see [6]. Also, note D is not antisymmetric, but is banded. D is not in itself the differentiation matrix, but D is the component of the differentiation matrix, $\mathcal{D} = C^{-1}DC$, which does the differencing. Later in this paper it will become obvious how one can simply "look" at D in order to tell the maximum differentiation accuracy at the boundaries. The next section begins the discussion of choosing a quadrature matrix C.

4. Approximating scaling function coefficients. Once again, let us work with only the D_4 wavelet. Recall, as constructed in [3], that there are two boundary functions at each end of the interval, as well as the usual D_4 scaling functions away from the boundaries. Let N denote the number of degrees of freedom of the approximation space V_0. Our function $f(x) \in L^2(R)$ is then discretized as follows. (In order to simplify the presentation I will let the "interval" be $[0, N]$, and there will be no overlap between left-hand boundary functions and right-hand boundary functions.)

First, the LHS,

$$s_i^L = \int_0^N \phi_i^L(x) f(x) dx, \tag{29}$$

then the right-hand side (RHS),

$$s_i^R = \int_0^N \phi_i^R(x) f(x) dx, \tag{30}$$

where $i = 0, 1$. Next, discretize in the middle with the usual scaling functions

$$s_k = \int_0^N \phi(x - k) f(x) dx \tag{31}$$

for $k = 2, \dots, N - 4$.

Now we will assume that $f(x)$ is defined on a discrete evenly spaced grid $f(x_i)$ for $i = 1, \dots, N$. Therefore, we must approximate the above three integrals by an appropriate quadrature matrix.

In the following two sections two types of quadrature matrices will be considered. The first type is constructed using the moments of the scaling function and is represented by

$$\vec{s} = C\vec{f}. \tag{32}$$

It will be seen that two-point quadrature matrices for this method are well conditioned, whereas three-point quadrature matrices are ill conditioned.

The second type of quadrature matrix is constructed from samples of the scaling function and boundary functions and is represented by

$$\vec{f} = \hat{C}\vec{s}. \tag{33}$$

It will be seen that this type of quadrature matrix is always ill conditioned.

5. Quadrature matrix C using moments. The quadrature matrix C maps from evenly spaced point values $f(x_i)$ of a function to the approximate scaling function coefficients at the finest scale $\vec{s}$. The quadrature matrix depends on three quantities: the number of vanishing moments of the wavelet, the grid that the function $f(x)$ is defined on, and which grid points are used to approximate each scaling function coefficient.

5.1. Moments of scaling and boundary functions. The moments of the D_4 scaling functions were calculated in Appendix A of [6]. The first three moments are $M_0 = 1$, $M_1 = .6339745962$, and $M_2 = .4019237886$.

We will concentrate here on calculating the moments of the boundary functions.

5.1.1. Moment 0 of LHS boundary functions. The 0th moment of the boundary function k is

$$\int_0^\infty \phi_k^L(x)dx = \sqrt{2}\sum_{l=0}^{1} h_{k,l}^L \int_0^\infty \phi_l^L(2x) + \sqrt{2}\sum_{m=2}^{2+2k} h_{k,m}^L \int_0^\infty \phi(2x-m). \tag{34}$$

Now, adopt the notation $m_k^0 = \int \phi_k^L(x)dx$ and let $y = 2x$ where appropriate in the above integrals to get

$$\sqrt{2}m_k^0 = \sum_{l=0}^{1} h_{k,l}^L m_l^0 + \sum_{m=2}^{2+2k} h_{k,m}^L \int_{-m}^\infty \phi(x)dx. \tag{35}$$

The integral $\int_{-m}^\infty \phi(x)dx$ is 1 since $m \geq 2$, and the scaling function, as defined here, is supported on $[-1, 2]$. Equation (35) defines the following 2-by-2 system:

$$\begin{bmatrix} h_{0,0}^L - \sqrt{2} & h_{0,1}^L \\ h_{1,0}^L & h_{1,1}^L - \sqrt{2} \end{bmatrix} \begin{bmatrix} m_0^0 \\ m_1^0 \end{bmatrix} = \begin{bmatrix} -h_{0,2}^L \\ -h_{1,2}^L - h_{1,3}^L - h_{1,4}^L \end{bmatrix}. \tag{36}$$

The solution to the above system is

$$m_0^0 = .3620520589,$$

$$m_1^0 = 1.001445402.$$

In a similar fashion all moments can be found for the LHS and RHS boundary functions.

5.2. The choice of grid points. In this subsection the affect of the choice of different grid points on the quadrature matrix will be illustrated. For simplicity, in this subsection we will work with only eight basis functions and eight grid points. The interval will be $[0, 8]$. When choosing our grid points to approximate the scaling function coefficient s_i we must use only the grid points that lie within the support of the basis function $b_i(x)$. That is, the inner product that is being approximated is

$$s_i = \int_0^8 b_i(x)f(x)dx,$$

and the data that is available to approximate s_i must be limited to function values defined within the support of $b_i(x)$. The support of each of the eight basis functions is

$$\begin{aligned} \text{supp}\{\phi_1^{bd}\} &= [0, 2], \\ \text{supp}\{\phi_2^{bd}\} &= [0, 3], \\ \text{supp}\{\phi_3\} &= [1, 4], \\ \text{supp}\{\phi_4\} &= [2, 5], \\ \text{supp}\{\phi_5\} &= [3, 6], \\ \text{supp}\{\phi_6\} &= [4, 7], \\ \text{supp}\{\phi_7^{bd}\} &= [5, 8], \\ \text{supp}\{\phi_8^{bd}\} &= [6, 8], \end{aligned}$$

where the superscript "*bd*" denotes a boundary function.

The following subsections will consider a few of the possible quadrature matrices corresponding to the following two grids:

$$\vec{x} = \begin{bmatrix} .5 \\ 1.5 \\ 2.5 \\ 3.5 \\ 4.5 \\ 5.5 \\ 6.5 \\ 7.5 \end{bmatrix} \tag{37}$$

and

$$\vec{y} = \begin{bmatrix} 0 \\ 8/7 \\ 2*8/7 \\ 3*8/7 \\ 4*8/7 \\ 5*8/7 \\ 6*8/7 \\ 8 \end{bmatrix}. \tag{38}$$

5.2.1. The entries of the quadrature matrices. Now, one must use the scaling function and boundary function moments and at least two grid points contained within the support of each basis function in order to build the quadrature matrix. The first row of the quadrature matrix will estimate the basis function coefficient for the boundary function with the smallest support ϕ_1^{bd}. Suppose we have chosen grid $\vec{x}$. We know that for the D_4 wavelet that constants and lines can be represented exactly in the wavelet subspace V_0. There are only two grid points within the support of ϕ_1^{bd}. Therefore, the solution to the following linear system will yield the two coefficients $c_{1,1}$ and $c_{1,2}$ in the first row of the quadrature matrix. The linear system is

$$\begin{bmatrix} \int_I \phi_1^{bd}(x)dx \\ \int_I \phi_1^{bd}(x)xdx \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ .5 & 1.5 \end{bmatrix} \begin{bmatrix} c_{1,1} \\ c_{1,2} \end{bmatrix}. \tag{39}$$

In a similar manner all the entries of a quadrature matrix can be found once one has calculated the moments of the basis functions and chosen the grid points. The following subsection will illustrate a few of the quadrature matrices associated with various grid choices.

5.2.2. All grid points contained in scaling function support. As a first possible choice of grid points, let us use every grid point that is contained within the support of the respective basis functions. Such a quadrature matrix will the following structure:

$$C_1^{3pt} = \begin{bmatrix} c_{1,1} & c_{1,2} & 0 & 0 & 0 & 0 & 0 & 0 \\ c_{2,1} & c_{2,2} & c_{2,3} & 0 & 0 & 0 & 0 & 0 \\ 0 & c_{3,2} & c_{3,3} & c_{3,4} & 0 & 0 & 0 & 0 \\ 0 & 0 & c_{4,3} & c_{4,4} & c_{4,5} & 0 & 0 & 0 \\ 0 & 0 & 0 & c_{5,4} & c_{5,5} & c_{5,6} & 0 & 0 \\ 0 & 0 & 0 & 0 & c_{6,5} & c_{6,6} & c_{6,7} & 0 \\ 0 & 0 & 0 & 0 & 0 & c_{7,6} & c_{7,7} & c_{7,8} \\ 0 & 0 & 0 & 0 & 0 & 0 & c_{8,7} & c_{8,8} \end{bmatrix}. \tag{40}$$

For both of the grids $\vec{x}$ and $\vec{y}$ the matrix C_1^{3pt} is ill conditioned. The condition number is proportional to N^2, where N is the number of grid points, and inversion of C_1^{3pt} unacceptably corrupts the differentiation matrix.

5.2.3. Two-point quadrature at boundary and three-point otherwise. The quadrature matrix has the following structure:

$$(41)\qquad C_2^{3pt} = \begin{bmatrix} c_{1,1} & c_{1,2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & c_{2,2} & c_{2,3} & 0 & 0 & 0 & 0 & 0 \\ 0 & c_{3,2} & c_{3,3} & c_{3,4} & 0 & 0 & 0 & 0 \\ 0 & 0 & c_{4,3} & c_{4,4} & c_{4,5} & 0 & 0 & 0 \\ 0 & 0 & 0 & c_{5,4} & c_{5,5} & c_{5,6} & 0 & 0 \\ 0 & 0 & 0 & 0 & c_{6,5} & c_{6,6} & c_{6,7} & 0 \\ 0 & 0 & 0 & 0 & 0 & c_{7,6} & c_{7,7} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & c_{8,7} & c_{8,8} \end{bmatrix}.$$

This matrix is also ill conditioned for both grids $\vec{x}$ and $\vec{y}$. Again, the condition number is proportional to N^2.

5.2.4. A first option using a two-point quadrature. We have

$$(42)\qquad C_1^{2pt} = \begin{bmatrix} c_{1,1} & c_{1,2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & c_{2,2} & c_{2,3} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & c_{3,3} & c_{3,4} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & c_{4,4} & c_{4,5} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & c_{5,5} & c_{5,6} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & c_{6,6} & c_{6,7} & 0 \\ 0 & 0 & 0 & 0 & 0 & c_{7,6} & c_{7,7} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & c_{8,7} & c_{8,8} \end{bmatrix}.$$

This quadrature matrix is well conditioned for both grids $\vec{x}$ and $\vec{y}$.

5.2.5. Second option for two-point quadrature. We have

$$(43)\qquad C_2^{2pt} = \begin{bmatrix} c_{1,1} & c_{1,2} & 0 & 0 & 0 & 0 & 0 & 0 \\ c_{2,1} & c_{2,2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & c_{3,3} & c_{3,4} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & c_{4,4} & c_{4,5} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & c_{5,5} & c_{5,6} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & c_{6,6} & c_{6,7} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & c_{7,7} & c_{7,8} \\ 0 & 0 & 0 & 0 & 0 & 0 & c_{8,7} & c_{8,8} \end{bmatrix}.$$

This quadrature matrix is also well conditioned for grids $\vec{x}$ and $\vec{y}$.

5.3. Conclusion. In this section a variety of structures for quadrature matrices have been shown where the matrices are found by using the moments of the scaling function. In general, the pattern is that three-point quadrature matrices are ill conditioned and two-point quadrature matrices are well conditioned.

The next section will explore quadrature matrices constructed from the samples of the scaling functions.

6. Quadrature matrix $\hat{C}$ using samples. Let $g(x)$ denote the projection of $f(x) \in L^2(R)$ in V_0:

$$(44)\qquad g(x) = P_{V_0} f(x) = \sum_{k=1}^{N} s_k b_k(x).$$

The quadrature formula based on samples of the basis functions can be found from

$$g(x_i) = P_{V_0} f(x_i) = \sum_{k=1}^{N} s_k b_k(x_i). \tag{45}$$

In matrix form the quadrature formula is

$$\vec{g} = \hat{C}\vec{s}, \tag{46}$$

or let $\vec{\sigma}$ be the approximation of $\vec{s}$ to get

$$\vec{f} = \hat{C}\vec{\sigma}, \tag{47}$$

where for an 8-by-8 case the quadrature matrix is

$$\hat{C} = \begin{bmatrix} \phi_0^L(x_1) & \phi_1^L(x_1) & 0 & 0 & 0 & 0 & 0 & 0 \\ \phi_0^L(x_2) & \phi_1^L(x_2) & \phi_2(x_2) & 0 & 0 & 0 & 0 & 0 \\ 0 & \phi_1^L(x_3) & \phi_2(x_3) & \phi_3(x_3) & 0 & 0 & 0 & 0 \\ 0 & 0 & \phi_2(x_4) & \phi_3(x_4) & \phi_4(x_4) & 0 & 0 & 0 \\ 0 & 0 & 0 & \phi_3(x_5) & \phi_4(x_5) & \phi_5(x_5) & 0 & 0 \\ 0 & 0 & 0 & 0 & \phi_4(x_6) & \phi_5(x_6) & \phi_1^R(x_6) & 0 \\ 0 & 0 & 0 & 0 & 0 & \phi_5(x_7) & \phi_1^R(x_7) & \phi_0^R(x_7) \\ 0 & 0 & 0 & 0 & 0 & 0 & \phi_1^R(x_8) & \phi_0^R(x_8) \end{bmatrix}. \tag{48}$$

Now $\hat{C}$ will be found for the two previously defined grids $\vec{x}$ and $\vec{y}$.

6.1. A quadrature matrix based on grid $\vec{x}$. Finding the explicit form of the above-defined $\hat{C}$ for the grid $\vec{x}$ is relatively simple since we only need the numerical values of the scaling functions and boundary functions halfway between the integers.

6.1.1. The numerical values of the scaling function. Using the definition of the scaling function supported on [0, 3],

$$\phi(x) = \sqrt{2}\sum_{k=0}^{3} h_k \phi(2x - k),$$

we get

$$\phi(5/2) = \sqrt{2}h_3\phi(2) = .06698729811, \tag{49}$$

$$\phi(3/2) = \sqrt{2}(h_1\phi(2) + h_2\phi(1)), \tag{50}$$

and

$$\phi(1/2) = \sqrt{2}h_0\phi(1) = .9330127019. \tag{51}$$

The numerical values of ϕ at the integers are provided in [9]:

$$\phi(1) = 1/2(1 + \sqrt{3})$$

and

$$\phi(2) = 1/2(1 - \sqrt{3}).$$

6.1.2. The numerical values of the LHS boundary functions. Given the numerical values of the scaling function $\phi(x)$ on grid $\vec{x}$ one can now find the numerical values of the boundary functions on grid $\vec{x}$ from the expression

$$\phi_k^L(x) = \sqrt{2}\sum_{l=0}^{1} h_{k,l}^L \phi_l^L(2x) + \sqrt{2}\sum_{m=2}^{2+2k} h_{k,m}^L \phi(2x-m). \tag{52}$$

In a similar manner one can find the numerical values for the RHS boundary functions to get the quadrature matrix of the next subsection.

6.1.3. The complete quadrature matrix for grid $\vec{x}$. An 8-by-8 example of the quadrature matrix for grid $\vec{x}$ is

$$\hat{C}_{\vec{x}} = \begin{bmatrix} .4124 & .8495 & 0 & 0 & 0 & 0 & 0 & 0 \\ .2062 & -.0077 & .9330 & 0 & 0 & 0 & 0 & 0 \\ 0 & .0669 & 0 & .9330 & 0 & 0 & 0 & 0 \\ 0 & 0 & .0670 & 0 & .9330 & 0 & 0 & 0 \\ 0 & 0 & 0 & .0670 & 0 & .9330 & 0 & 0 \\ 0 & 0 & 0 & 0 & .0670 & 0 & .8561 & 0 \\ 0 & 0 & 0 & 0 & 0 & .0670 & .3270 & .4451 \\ 0 & 0 & 0 & 0 & 0 & 0 & -.1406 & .8902 \end{bmatrix}. \tag{53}$$

This matrix is ill conditioned just as the three-point quadrature formulas were for the quadrature matrix derived using moments.

6.2. A quadrature matrix based on grid $\vec{y}$. Using the definition of the scaling function

$$\phi(x) = \sqrt{2}\sum_{k=0}^{3} h_k \phi(2x-k),$$

one can set up a matrix such that the eigenvector for the eigenvalue $\lambda = 1$ gives a scalar multiple of the scaling function $\phi(x)$ at $x = i/7$ for $i = 1, \ldots, 20$. The eigenspace for the eigenvalue $\lambda = 1$, however, is two dimensional. To avoid this complication one can approximate the numerical values of $\phi(x)$ at $x = i/7$ for $i = 1, \ldots, 20$ by generating $\phi(x)$ at a large number of diadic points; in this case 2^{14} points were chosen. This approximation is sufficient to understand the character of the quadrature matrix, specifically if it is well conditioned or not. Once the numerical values of the scaling function are known then we can find the numerical values of the boundary functions.

6.2.1. The numerical values of the LHS boundary functions. As before, the numerical values of the LHS boundary functions on grid $\vec{y}$ can be found from

$$\phi_k^L(x) = \sqrt{2}\sum_{l=0}^{1} h_{k,l}^L \phi_l^L(2x) + \sqrt{2}\sum_{m=2}^{2+2k} h_{k,m}^L \phi(2x-m).$$

The numerical values in terms of the scaling function values on grid $\vec{y}$ are

$$\begin{aligned} &\phi_0^L(8/7) = \sqrt{2}(h_{0,1}^L \phi_1^L(16/7) + h_{0,2}^L \phi(2/7)), \\ &\phi_1^L(8/7) = \sqrt{2}(h_{1,1}^L \phi_1^L(16/7) + h_{1,2}^L \phi(2/7) + h_{1,3}^L \phi(-5/7)), \\ &\phi_1^L(16/7) = \sqrt{2}(h_{1,3}^L \phi(11/7) + h_{1,4}^L \phi(4/7)). \end{aligned} \tag{54}$$

As with grid $\vec{x}$ the quadrature matrix produced using grid $\vec{y}$ is ill conditioned. Recall from the previous section that all three-point quadrature matrices were ill conditioned. Therefore, I conclude this section with the observation, not a proof, that quadrature matrices for wavelet bases on an interval using the construction in [3] should be constructed using moments and that the number of grid points used to approximate each scaling function coefficient should be equal to the number of vanishing moments of the corresponding wavelet.

7. Order of accuracy. In this section the order of accuracy for a few of the differentiation matrices that can be constructed from the derivative projection matrix D and the quadrature matrices constructed in the previous three sections will be found. We begin with the differentiation matrices constructed using grid $\vec{x}$ followed by the differentiation matrices constructed using grid $\vec{y}$.

7.1. Accuracy using grid $\vec{x}$. Recall that grid $\vec{x}$ is comprised of the points halfway between the integers and, consequently, the spacing between the grid points is the same as the translation distance for the scaling functions on the finest scale. Furthermore, this grid does not provide a grid point at the boundary itself. This lack of a grid point at the boundary makes this grid an unlikely choice, but I have chosen to study the accuracy as an experiment. Let us begin by giving a few explicit examples of differentiation matrices.

7.1.1. Differentiation matrix $(C_1^{2pt})^{-1}DC_1^{2pt}$, grid $\vec{x}$. Note the unusual form of the following matrix. It is essentially upper triangular in form. This form is due to the inversion of C_1^{2pt}. From Table 1 it can be seen that this differentiation matrix does not maintain the superconvergence at the boundaries. That is, the differentiation is first-order accurate at the boundaries, whereas it is fourth-order accurate at the middle 50% of the grid points. The total differentiation accuracy across the entire domain, therefore, is second-order accurate. We have

$$\mathcal{D} =$$

$$\begin{bmatrix}
-25.33 & 42.38 & -20.77 & 3.73 & -6e^{-4} & 0 & 0 & -7e^{-4} & 7e^{-3} & -9e^{-3} & 2e^{-5} & 3e^{-3} \\
-5.71 & -1.59 & 8.35 & -1.04 & -1e^{-3} & 0 & 0 & -2e^{-3} & .02 & -.02 & 3e^{-5} & 6e^{-3} \\
1.05 & -8.11 & .05 & 8 & -1 & 0 & 0 & -3e^{-3} & .03 & -.04 & 7e^{-5} & .01 \\
0 & 1 & -8 & 0 & 8 & -1 & 0 & -7e^{-3} & .07 & -.09 & 2e^{-4} & .03 \\
0 & 0 & 1 & -8 & 0 & 8 & -1 & -.02 & .16 & -.20 & 3e^{-4} & .06 \\
0 & 0 & 0 & 1 & -8 & 0 & 8 & -1.03 & -.34 & -.43 & 7e^{-4} & .13 \\
0 & 0 & 0 & 0 & 1 & -8 & 0 & 7.93 & -.28 & -.94 & 2e^{-3} & .29 \\
0 & 0 & 0 & 0 & 0 & 1 & -8 & -.16 & 9.56 & -3.02 & 3e^{-3} & .62 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -8.35 & 3.36 & 3.65 & -.99 & 1.33 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .25 & -.76 & -8.37 & 6.02 & 2.86 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1.62 & 16.61 & -33.66 & 11.96 & 6.71 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .53 & -5.47 & 2.78 & -3.27 & 5.43
\end{bmatrix}.$$

7.1.2. Differentiation matrix $(C_2^{2pt})^{-1}DC_2^{2pt}$, grid $\vec{x}$. Now we use the quadrature matrix C_2^{2pt}. But the results are similar to the previous matrix. The structure is unacceptable and the superconvergence is lost at the boundaries as can be seen from Table 1. We have

$$\mathcal{D} =$$

$$\begin{bmatrix}
-12.21 & 7.44 & 10.48 & -6.44 & .74 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-10.99 & -11.15 & 44.98 & -24.53 & 1.70 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-2.50 & -1.01 & -3.50 & 8 & -1 & 0 & 0 & 0 & 0 & .01 & -.01 & 0 \\
.46 & .07 & -7.54 & 0 & 8 & -1 & 0 & 0 & 0 & .01 & -.02 & .01 \\
0 & 0 & 1 & -8 & 0 & 8 & -1 & 0 & -.03 & .02 & -.04 & .02 \\
0 & 0 & 0 & 1 & -8 & 0 & 8 & -1 & -.01 & .05 & -.09 & .04 \\
0 & 0 & 0 & 0 & 1 & -8 & 0 & 8 & -1.01 & .11 & -.19 & .09 \\
0 & 0 & 0 & 0 & 0 & 1 & -8 & 0 & 7.97 & -.75 & -.41 & .19 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -8 & -.06 & 8.53 & -1.89 & .04 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -8.13 & 1.14 & 6.09 & -.11 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & .73 & -5.53 & -3.12 & 7.93 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -.24 & 2.74 & -16.75 & 14.26
\end{bmatrix}.$$

TABLE 1

Differentiation accuracy for Daubechies wavelets defined on an interval using grid points halfway between the integers.

Differentiation matrix	Grid size	Boundary l_1 error	Error ratio	Inner l_1 error	Error ratio	Total l_1 error	Error ratio
$(C_1^{2pt})^{-1}DC_1^{2pt}$	16	.0066		$2.86e^{-4}$		.0024	
Grid $\vec{x}$	32	.0032	2.06	$3.20e^{-6}$	89	$5.73e^{-4}$	4.19
	64	.0016	2.00	$2.10e^{-9}$	1524	$1.41e^{-4}$	4.06
	128	$7.85e^{-4}$	2.04	$2.80e^{-11}$	75	$3.50e^{-5}$	4.03
	256	$3.81e^{-4}$	2.06	$1.75e^{-12}$	16.0	$8.53e^{-6}$	4.10
	512	$1.69e^{-4}$	2.25	$1.07e^{-13}$	16.4	$1.94e^{-6}$	4.40
$(C_2^{2pt})^{-1}DC_2^{2pt}$	16	.0091		$2.71e^{-5}$		.0026	
Grid $\vec{x}$	32	.0047	1.94	$3.13e^{-7}$	86.58	$6.78e^{-4}$	3.83
	64	.0024	1.96	$5.92e^{-10}$	529	$1.72e^{-4}$	3.94
	128	.0012	2.00	$2.80e^{-11}$	21.1	$4.33e^{-5}$	3.97
	256	$6.08e^{-4}$	1.97	$1.75e^{-12}$	16.0	$1.10e^{-5}$	3.94
	512	$3.22e^{-4}$	1.89	$1.07e^{-13}$	16.4	$2.89e^{-6}$	3.81
$\hat{C}_{\vec{x}}DC_1^{3pt}$	16	.0038		$6.52e^{-5}$		.0013	
Grid $\vec{x}$	32	.0019	2.00	$1.60e^{-5}$	4.08	$3.15e^{-4}$	4.13
	64	$9.23e^{-4}$	2.06	$3.95e^{-6}$	4.05	$7.87e^{-5}$	3.99
	128	$4.64e^{-4}$	1.99	$9.83e^{-7}$	4.02	$1.97e^{-5}$	3.99
	256	$2.32e^{-4}$	2.00	$2.45e^{-7}$	4.01	$4.92e^{-6}$	4.00
	512	$1.18e^{-4}$	1.97	$6.12e^{-8}$	4.00	$1.22e^{-6}$	4.04

7.1.3. Conclusion for grid $\vec{x}$. Recall that in the construction of the differentiation matrix that the middle matrix D is fixed. Table 1 contains the accuracy for two well-conditioned quadrature matrices C_1^{2pt} and C_2^{2pt}. In addition, as an experiment I have combined two ill-conditioned matrices: one mapping from $\vec{f}$ to $\vec{s}$ and the other mapping from $\vec{s}$ to $\vec{f}$.

First the noteworthy points are that in all three cases the accuracy at the boundary is first order. Recall that when periodic boundary conditions are imposed that the differentiation accuracy is fourth order for the D_4 wavelet. This loss of superconvergence is a serious problem. Furthermore, note from the table that the two well-conditioned quadrature matrices maintain the superconvergence away from the boundary. But in all three cases the total error of the differentiation matrix across the entire domain is one higher than the boundary accuracy; i.e., we get second-order differentiation accuracy.

Note: Inner accuracy is calculated at the middle 50% of the grid points, and boundary accuracy is calculated at the two outermost grid points at each end of the interval.

7.2. Accuracy using grid $\vec{y}$. We now study the differentiation accuracy where we have included a boundary point at each end of the interval in the grid and the remainder of the grid points are required to be evenly spaced across the interval. This choice of grid is a much more likely choice than grid $\vec{x}$. However, superconvergence again is lost at the boundary. This reduces the total accuracy of the differentiation to second order as with grid $\vec{x}$. Again, let us note a few explicit examples of differentiation matrices.

7.2.1. Differentiation matrix $(C_1^{2pt})^{-1}DC_1^{2pt}$, grid $\vec{y}$. Again note the unusual, and undesirable, form of the following matrix. As can be seen from Table 2, again, the superconvergence is lost at the boundary. We have

$$\mathcal{D} =$$

$$\begin{bmatrix}
-1.99 & 3.33 & -1.60 & .26 & -2e^{-4} & -1e^{-4} & -6e^{-5} & -4e^{-5} & 1e^{-4} & -2e^{-4} & 6e^{-5} & 2e^{-5} \\
-.46 & -.14 & .72 & -.13 & -1e^{-3} & -6e^{-4} & -3e^{-4} & -2e^{-4} & 7e^{-4} & -1e^{-3} & 3e^{-4} & 1e^{-4} \\
.08 & -.63 & -.02 & .70 & -.12 & -2e^{-3} & -1e^{-3} & -7e^{-4} & 2e^{-3} & -4e^{-3} & 1e^{-3} & 3e^{-4} \\
0 & .07 & -.63 & -.02 & .69 & -.12 & -3e^{-3} & -2e^{-3} & 7e^{-3} & -.01 & 3e^{-3} & 1e^{-3} \\
0 & 0 & .08 & -.63 & -.02 & .69 & -.11 & -5e^{-3} & .02 & -.03 & 9e^{-3} & 3e^{-3} \\
0 & 0 & 0 & .08 & -.63 & -.02 & .69 & -.12 & .05 & -.08 & .02 & 7e^{-3} \\
0 & 0 & 0 & 0 & .08 & -.63 & -.02 & .68 & 5e^{-3} & -.18 & .05 & .02 \\
0 & 0 & 0 & 0 & 0 & .08 & -.63 & -.04 & .94 & -.50 & .12 & .03 \\
0 & 0 & 0 & 0 & 0 & 0 & .08 & -.69 & .52 & -.12 & .15 & .07 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -.03 & .43 & -1.54 & 1.00 & .14 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -.20 & 2.12 & -4.25 & 2.05 & .29 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .44 & -4.62 & 7.00 & -2.83 & 8e^{-3}
\end{bmatrix}.$$

It has been mentioned many times that the structure is undesirable. This is due to the lack of directional independence of the matrix. One "fix" for this problem is given in the next subsection.

7.2.2. The no-preferred-direction (NPD) differentiation matrix $(C_1^{2pt})^{-1}DC_1^{2pt}$, grid $\vec{y}$. It is possible to fix one of the problems with the above matrices: the problem of directional dependence. That is, a general differentiation matrix should never treat data coming from the right differently than data coming from the left. This problem can be "fixed" if one simply turns the basis around and builds another differentiation matrix. That is, turn all the inner scaling functions around and interchange the role of the RHS and LHS boundary functions. The new differentiation matrix $\tilde{\mathcal{D}}$ can be found from the unreversed differentiation matrix $\mathcal{D}$ by first flipping the matrix by a middle horizontal axis and then by a middle vertical axis and then negating. That is,

$$\tilde{\mathcal{D}} = -flipud(fliplr(\mathcal{D})), \tag{55}$$

and we build a new NPD differentiation matrix by averaging these two matrices:

$$\mathcal{D}_{NPD} = \frac{1}{2}(\mathcal{D} + \tilde{\mathcal{D}}). \tag{56}$$

TABLE 2

Differentiation accuracy for Daubechies wavelets defined on an interval using grid points at the boundaries and all other grid points evenly spaced across the interval.

Differentiation matrix	Grid size	Boundary l_1 error	Error ratio	Inner l_1 error	Error ratio	Total l_1 error	Error ratio
$(C_1^{2pt})^{-1}DC_1^{2pt}$	16	.0181		$8.70e^{-4}$		.00598	
grid $\vec{y}$	32	.00933	1.94	$2.31e^{-4}$	3.77	.00162	3.69
	64	.00474	1.96	$6.04e^{-5}$	3.82	$4.26e^{-4}$	3.80
	128	.00238	1.99	$1.49e^{-5}$	4.05	$1.089e^{-4}$	3.91
	256	.00117	2.03	$3.73e^{-6}$	3.99	$2.72e^{-5}$	4.00
NPD of	16	.0211		$6.82e^{-4}$		.00692	
$(C_1^{2pt})^{-1}DC_1^{2pt}$	32	.0116	1.82	$1.42e^{-5}$	48.0	.00191	3.62
grid $\vec{y}$	64	.00606	1.91	$5.88e^{-6}$	2.41	$5.05e^{-4}$	3.78
	128	.00310	1.95	$1.43e^{-6}$	4.11	$1.30e^{-4}$	3.88
	256	.00156	1.99	$3.50e^{-7}$	4.09	$3.29e^{-5}$	3.95

The following NPD matrix treats data coming from the left in the same manner as data coming from the right. Note that this matrix is full.

$$\mathcal{D}_{NPD} =$$

$$\begin{bmatrix}
-12.00 & 36.96 & -51.58 & 29.30 & -2.67 & -7e^{-4} & -3e^{-4} & -2e^{-4} & 8e^{-4} & -1e^{-3} & 4e^{-5} & 1e^{-4} \\
-4.47 & -13.12 & 29.85 & -13.47 & 1.22 & -4e^{-3} & -2e^{-3} & -1e^{-3} & 4e^{-3} & -7e^{-3} & 2e^{-3} & 6e^{-4} \\
-.39 & -9.77 & 9.11 & 1.62 & -.54 & -.01 & -6e^{-3} & -4e^{-3} & .01 & .-.02 & 7e^{-3} & 2e^{-3} \\
-.43 & -.43 & -3.01 & -3.24 & 8.30 & -1.16 & -.02 & -.01 & .04 & -.07 & .02 & 6e^{-3} \\
-.20 & -.70 & 3.44 & -9.44 & .12 & 7.96 & -1.15 & -.03 & .11 & -.18 & .06 & .02 \\
-.09 & -.32 & 1.09 & .42 & -7.86 & -9e^{-3} & 7.94 & -1.16 & .28 & -.47 & .14 & .04 \\
-.04 & -.14 & .47 & -.28 & 1.16 & -7.94 & 9e^{-3} & 7.86 & -.42 & -1.09 & .32 & .09 \\
-.02 & -.06 & .19 & -.11 & .03 & 1.15 & -7.96 & -.12 & 9.44 & -3.44 & .70 & .20 \\
-6e^{-3} & -.02 & .07 & -.04 & .01 & .02 & 1.16 & -8.30 & 3.24 & 3.01 & .43 & .43 \\
-2e^{-3} & -7e^{-3} & .02 & -.01 & 4e^{-3} & 6e^{-3} & .01 & .54 & -1.62 & -9.11 & 9.77 & .39 \\
-6e^{-4} & -2e^{-3} & 7e^{-3} & -4e^{-3} & 1e^{-3} & 2e^{-3} & 4e^{-3} & -1.22 & 13.47 & -29.85 & 13.11 & 4.47 \\
-1e^{-4} & -4e^{-4} & 1e^{-3} & -8e^{-4} & 2e^{-4} & 3e^{-4} & 7e^{-4} & 2.67 & -29.29 & 51.58 & -36.96 & 12.00
\end{bmatrix}.$$

Table 2 gives the accuracy results for two differentiation matrices constructed from the well-conditioned quadrature matrix C_1^{2pt}.

Again, Table 2 shows the loss of superconvergence at the boundaries regardless of the differentiation matrix. Also, note that for grid $\vec{y}$ the superconvergence is not even achieved away from the boundary as it is for grid $\vec{x}$. Inner error and boundary error are defined in the same manner as for the previous table.

In conclusion, in this section I have presented a number of differentiation matrices along with the orders of accuracy of the differentiation. In all cases the superconvergence that is encountered under periodic boundary conditions is lost.

8. Other constructions and conclusion. In this paper I have explored the differentiation matrix for Daubechies-type wavelets defined on an interval using the boundary construction defined in [3]. The exploration has been limited to the wavelet with two vanishing moments usually referred to as the D_4 wavelet. It was seen that the superconvergence encountered with periodic boundary conditions which was proven in [6] is lost with this boundary construction.

There are other boundary constructions available for Daubechies wavelets, see [1], [8], but none of the constructions can maintain the superconvergence. There is a very straightforward way to find the maximum accuracy at the boundary for a given boundary construction. That is, when the basis functions are orthogonal under translation, as they are for Daubechies wavelets, one can simply count the number of basis functions which have support overlapping the support of the outermost boundary functions. This defines the number of nonzero entries in the first and last rows of the middle matrix D referred to here as the derivative projection matrix. All differencing in wavelet differentiation is done by D, and, therefore, the differentiation accuracy at the boundaries cannot exceed one less than the number of nonzero entries in the first and last rows of D. Of course, this argument holds for every row of D, but I am particularly concerned with the boundaries here.

Perhaps it is not possible to construct boundary functions for wavelets which give everything. That is, boundary functions which maintain the wavelet structure, which maintain the approximation properties across the entire interval, and which maintain superconvergence. This conjecture is based first on the fact that to date there is no boundary construction which satisfies these criteria for either Daubechies wavelets or spline-based wavelets; see [7]. In further support of this conjecture, in a paper by Gottlieb et al. [5], it is proven that it is not possible to construct boundary functions for a finite element subspace which maintain the superconvergence encountered with periodic boundary conditions if there are characteristics leaving the domain. Note that the finite element subspace is the same as the first-order spline wavelet subspace; see [7].

In conclusion, under the currently available boundary constructions the superconvergence, which is encountered under periodic boundary conditions, is not maintained at the boundaries. In the case of the D_4 wavelet under the boundary construction examined here one gets fourth-order differentiation away from the boundaries and only first-order differentiation at the boundaries producing a scheme with only second-order accurate spatial differentiation. It remains to be seen if it is possible to construct boundary functions for a wavelet subspace which can maintain superconvergence at the boundaries.

REFERENCES

[1] P. Auscher, *Wavelets with boundary conditions on the interval*, in Wavelets: A Tutorial in Theory and Applications, C. K. Chui, ed., Academic Press, New York, 1992, pp. 217-236.

[2] G. Beylkin, *On the representation of operators in bases of compactly supported wavelets*, SIAM J. Numer. Anal., 29 (1992), pp. 1716-1740.

[3] A. Cohen, I. Daubechies, and P. Vial, *Wavelets on the interval and fast wavelet transforms*, Appl. Computational Harmonic Analysis, 1 (1993), pp. 54-81.

[4] I. Daubechies, *Orthonormal basis of compactly supported wavelets*, Comm. Pure Appl. Math., 41 (1988), pp. 909-996.

[5] D. Gottlieb, B. Gustafsson, P. Olsson, and B. Strand, *On the superconvergence of Galerkin methods for hyperbolic IBVP*, SIAM J. Numer. Anal., to appear.

[6] L. Jameson, *On the wavelet based differentiation matrix*, J. Sci. Comput., September 1993.

[7] ———, *On the spline-based wavelet differentiation matrix*, J. Appl. Numer. Math., 17 (1995), pp. 33-45.

[8] Y. Meyer, *Ondelettes sur i'ntervalle*, Rev. Mat. Iberoamericana, 7 (1992), pp. 115-133.

[9] G. Strang, *Wavelets and dilation equations: A brief introduction*, SIAM Rev., 31 (1989), pp. 614-627.

[10] M. Unser and A. Aldroubi, *Polynomial splines and wavelets—A signal processing perspective*, in Wavelets—A Tutorial in Theory and Applications, C. K. Chui, ed., Academic Press, Inc., New York, 1992, pp. 91-112.

SIAM J. SCI. COMPUT.
Vol. 17, No. 2, pp. 517-537, March 1996

A DATA SMOOTHING TECHNIQUE FOR PIECEWISE CONVEX/CONCAVE CURVES*

W. LI†, D. NAIK†, AND J. SWETITS†

Abstract. A data smoothing technique for piecewise convex/concave curves is developed and its computational aspects are discussed. Some of its statistical properties are determined and the difficulty in computing the general expression for bias and the mean squared error are illustrated through an example. A variety of examples are considered to demonstrate the potential of the proposed smoothing technique, and the behavior of the piecewise convex/concave smoother is shown by Monte-Carlo simulations. We also give a detailed comparison between the proposed smoothing technique and spline smoothing.

Key words. nonparametric regression, divided difference, monotone regression, convex regression, quadratic program, piecewise convex/concave curve, spline smoothing, Monte-Carlo simulation, mean squared error, bias

AMS subject classifications. 62G99, 90C90, 65D10

1. Introduction.

Given a set of sample data $\{(x_i, y_i) : 1 \leq i \leq n\}$, where $y_i = f(x_i)+\eta_i$ with some piecewise monotone or convex/concave function $f(x)$ on a closed interval $[a, b]$ and η_i is the random noise induced by the sampling process, we want to estimate the original function $f(x)$. Classical examples of regression by piecewise monotone or convex/concave functions are monotone regression and convex regression, which are two important cases in regression under order restriction. In certain applications, the number of extreme points or inflection points of the underlying function is very important for any useful estimator of $f(x)$. For example, in the processing of a digitized image of a profile curve of a mechanical part, one can have an a priori estimate of how many inflection points the curve has. As a consequence, one expects to have a curve with the same number of inflection points as an approximation of the profile curve. Levels of certain hormones in a woman are to go up and down during a one month period. Therefore, it is a practical need to model the hormone level by a function that has one inflection point in a month. For the problem of estimating the hazard rate of change, one wants to find the location of the extrema of the derivative $f'(x)$ (or the maximum jump of $f'(x)$). Müller and Wang [30], [31] proposed the following nonparametric approach for solving the change-point problem: (1) find a nonparametric estimator $\hat{f}(x)$ of $f(x)$ from the given sample; (2) use the change-point of $\hat{f}(x)$ as an estimator of the change-point for $f(x)$. They used this model to analyze life expectancy of 65 patients with acute lymphocytic leukemia after bone marrow transplantation. Note that if one knows that $f'(x)$ has one extreme point, then it is wise to use an $\hat{f}(x)$ that has only one inflection point as an estimator of $f(x)$. In fact, a recent study by Cuevas and Gonzalez-Manteiga [4] revealed that the weak convergence of a sequence of density functions $\{f_n(x)\}$ is equivalent to the strong convergence of $\{f_n(x)\}$ if all density functions have the same number of inflection points. Moreover, under certain conditions, the inflection points of $f_n(x)$ converge to the inflection points of $f(x)$. Roughly speaking, forcing constraint on the number of inflection points yields more reliable estimators. As pointed out by Cuevas and Gonzalez-Manteiga [4], an interesting consequence is that the bumps and the dips (i.e., the intervals of concavity and convexity) of $f(x)$ can be estimated consistently from those of $f_j(t)$; therefore, an inflection-point-preserving regression method is well suited for the so-called bump-hunting problem [16], [39], [32]. There are both practical and theoretical reasons for finding an estimator of $f(x)$ that resembles $f(x)$ geometrically. In this paper, we propose an efficient algorithm for fitting $f(x)$ by an estimator $\hat{f}(x)$ that has an a priori given number of inflection points.

*Received by the editors November 2, 1992; accepted for publication (in revised form) October 31, 1994.

†Department of Mathematics and Statistics, Old Dominion University, Norfolk, VA 23529.

The geometric shape parameters for a piecewise monotone or convex/concave function $f(x)$ are given by

$$I(g) := \inf \{m : \text{there exists a partition of } [a,b] \text{ into } m \text{ intervals } T_1, \dots, T_m \text{ such that } g(x) \text{ is convex or concave on every } T_i \text{ for } i = 1, \dots, m\}$$

and

$$M(g) := \inf \{m : \text{there exists a partition of } [a,b] \text{ into } m \text{ intervals } T_1, \dots, T_m \text{ such that } g(x) \text{ is monotone on every } T_i \text{ for } i = 1, \dots, m\}.$$

The integer $I(g)$ behaves like the number of inflection points of $g(x)$, and $M(g)$ like the number of relative extrema of $g(x)$. Let $\mathcal{I}_m := \{g(x) : g(x) \in \mathcal{W} \text{ and } I(g) = m\}$ and $\mathcal{M}_m := \{g(x) : g(x) \in \mathcal{W} \text{ and } M(g) = m\}$, where $\mathcal{W}$ is a given set of functions. Then, consider least squares fitting with constraints on $I(g)$ or $M(g)$:

$$\min_{g \in \mathcal{M}_m} \sum_{i=1}^{n} (g(x_i) - y_i)^2 \tag{1}$$

or

$$\min_{g \in \mathcal{I}_m} \sum_{i=1}^{n} (g(x_i) - y_i)^2, \tag{2}$$

where m is a given positive integer which controls the fitting scheme and $\mathcal{W}$ represents either a parametric or a nonparametric model. Let

$$S_n := \{g(x) : g(x) \text{ is a linear spline with knots } x_1 < x_2 < \cdots < x_n\},$$

i.e., $g(x) \in S_n$ if and only if $g(x)$ is a continuous function on $[x_1, x_n]$ and is a linear polynomial on each subinterval $[x_{i-1}, x_i]$. For $\mathcal{W} = S_n$, (1) and (2) are the piecewise monotone and convex/concave data smoothing methods, respectively, proposed by Demetriou and Powell [9], [7]. Here we consider $S_n \equiv \mathbf{R}^n$, the n-dimensional vector space, since every $g(x)$ in S_n is uniquely determined by its values at $x_1, \dots, x_n$. In particular, for $\mathcal{W} = S_n$ and $m = 1$, (1) is the classical monotone regression problem and is considered an important case of statistical inference under ordered restrictions. The monotone regression problem has been thoroughly investigated and documented [1], [35]. For $\mathcal{W} = S_n$ and $m = 1$, (2) is the classical convex/concave regression problem which was first introduced by Hildreth [21] for the estimation of production functions. Twenty two years after that, Hanson and Pledger [17] established the consistency of concave regression estimators while Demetriou and Powell [10] provided an algorithm suited specifically to this problem. Some more applications of convex/concave regression can be found in [28], [27], and [8]. Even though the convex/concave regression problem is a standard strictly convex quadratic programming problem, the ill-conditioned nature of the second divided difference matrix could cause some difficulty for efficient computation of the solution [25], [10], [5], [28], [24]. Moreover, even if $\mathcal{W} = S_n$, neither $\mathcal{M}_m$ nor $\mathcal{I}_m$ with $m \geq 2$ is a polyhedral set. In fact, they are not convex sets. Hence it becomes extremely difficult to find a numerical solution to (1) or (2). In a recent paper, Demetriou and Powell [9] developed a dynamic programming procedure with complexity $\mathcal{O}(n^2 + mn \ln n)$ for solving (1) with $\mathcal{W} = S_n$. However, for $m > 2$, (2) presents a challenging computational problem (cf. [22], [6] and Chap. 6 of [7]).

While (1) or (2) provides a practical means to compute a regression estimator with certain geometric shape from the sample data $\{(x_i, y_i)\}$, it is important to ensure the consistency of piecewise monotone (or convex/concave) regression estimators. Motivated by Holm and Frisén's work, Mammen [26] considered the following regression problem:

$$(3) \qquad \min_{g \in \mathcal{H}_{k,m,D}} \sum_{i=1}^{n} (g(x_i) - y_i)^2,$$

where, for $k \geq 1$,

$$\mathcal{H}_{k,m,D} := \left\{ g(x) : I\left(g^{(k-1)}\right) \leq m \text{ and there exists a scalar } \gamma \right.$$
$$\left. \text{such that } \sup_{x_1 \neq x_2} \left| \frac{g^{(k-1)}(x_1) - g^{(k-1)}(x_2)}{x_1 - x_2} - \gamma \right| \leq D \right\}$$

and

$$\mathcal{H}_{0,m,D} := \left\{ g(x) : M(g) \leq m \text{ and } \sup_x g(x) - \inf_x g(x) \leq D \right\}.$$

Mammen [26] proved that, for every function $g(x) \in \mathcal{H}_{k,m,D}$, there exists a spline function $\hat{g}(x)$ of order k such that $g(x_i) = \hat{g}(x_i)$ for $1 \leq i \leq n$ and $\hat{g}^k(x)$ is a piecewise constant function. Mammen [26] also established an asymptotic stochastic bound for the distance between the regression function $f(x)$ and the $\mathcal{H}_{k,m,D}$ least squares estimator. However, computationally, (3) seems to be more difficult than (2). In fact, Mammen [26] suggested considering $g(x)$ as a function defined on the design points $\{x_1, \ldots, x_n\}$ and replace the $(k-1)$th derivative by divided difference of order $(k-1)$. With such a discretization for $k = 1$ and $m = 2$, Mammen [26] uses a successive projection method introduced by Dykstra [13] (cf. also [11] and [12]) ($(n-2)$ times!) to find the corresponding regression estimator and compares it with a kernel estimator by simulations for two regression functions on [0,1].

Our research interest on piecewise convex/concave regression was motivated by the k-convex regression problem [5], [24]:

$$(4) \qquad \min \sum_{i=1}^{n} (\theta_i - y_i)^2$$

subject to the following constraints on the kth divided difference of $\{\theta_i\}_1^n$:

$$(5) \qquad \Delta_k \theta \geq 0,$$

where Δ_k is the kth divided difference matrix defined as follows:

$$(\Delta_k \theta)_j = \sum_{i=j}^{k+j} \left(\theta_i \prod_{\substack{s=j \\ s \neq i}}^{k+j} (x_i - x_s)^{-1} \right), \quad j = 1, \ldots, n-k.$$

However, due to the stringent constraints (5), the mean least squared error $\frac{1}{n} \sum_{i=1}^{n} (\theta_i - f(x_i))^2$ is usually too large in comparison with the noise level. Therefore, we want to relax the constraints and find a method which can produce a curve with a given number of convex/concave pieces. For convenience, we will identify a vector $\theta := (\theta_1, \ldots, \theta_n)$ with its unique spline interpolator $\theta(x)$ in S_n so that $I(\theta)$ and $M(\theta)$ can be defined for vectors $(\theta_1, \ldots, \theta_n)$. With such a convention, we propose the following nonparametric data smoothing method:

$$\min_{-\epsilon \le \Delta_k \theta \le \epsilon, I(\theta)=m} \sum_{i=1}^{n} (\theta_i - y_i)^2, \tag{6}$$

where ϵ is a nonnegative scalar. One could consider (6) as a discretized version of least squares fitting by functions whose kth derivative is bounded by ϵ:

$$\min_{\theta \in \mathcal{B}_\epsilon, I(\theta)=m} \sum_{i=1}^{n} (\theta(x_i) - y_i)^2, \tag{7}$$

where $\mathcal{B}_\epsilon := \{g(x) : |g^{(k)}(x)| \le \epsilon\}$. Note that (7) is actually (2) with $\mathcal{W} \equiv \mathcal{B}_\epsilon$. When $k = 1$, $\{g(x) : |g^{(k)}(x)| \le \epsilon, I(g) = m\}$ is more or less the same set as $\mathcal{H}_{1,m,\epsilon}$. In a sense, (6) is a variation of (3). However, as we mentioned before, (6) is a computationally difficult problem even without the constraints on the kth divided difference of θ. Therefore, we replace (6) by the following least squares problem:

$$\min_{-\epsilon \le \Delta_k \theta \le \epsilon} \sum_{i=1}^{n} (\theta_i - y_i)^2, \tag{8}$$

where ϵ controls the smoothing process. However, we still want to produce an estimator with m convex/concave pieces by an appropriate choice of ϵ. Note that (8) is actually a regression by polynomials of degree at most $(k-1)$ when $\epsilon = 0$. In general, the estimator is smoother if ϵ is smaller. On the other hand, the least squared error will be reduced if ϵ is increased and is zero if ϵ is large enough. Therefore, in order to make a good fit to the sample data, we want to make ϵ as large as possible; meanwhile, we want to control the magnitude of ϵ to produce a smooth estimator. Hence, it seems a good idea to choose the largest ϵ such that $I(\hat{\theta}(\epsilon)) = m$ for the solution $\hat{\theta}(\epsilon)$ to (8). Such a strategy for the selection of ϵ reflects the following concern about the selection of a smoothing parameter. Theoretically, under certain assumptions, one can establish some kinds of asymptotic results to justify a particular strategy for selection of the smoothing parameter (or the bandwidth) for a nonparametric regression method. However, for practical applications, the choice of the smoothing parameter is mainly a delicate balance between the fidelity to the sample data and the visual smoothness of the fitting curve. Based on the consistency results established by Cuevas and Gonzalez-Manteiga and by Mammen, it is a statistically sound decision to choose the largest ϵ such that $I(\hat{\theta}(\epsilon)) = m$. Our strategy for selection of the smoothing parameter is very similar to Silverman's bandwidth selection for a kernel smoothing method [37], [38]. See §2 for more details.

For a given ϵ, one can find $\hat{\theta}(\epsilon)$ by any algorithm designed for strictly convex quadratic programming problems. However, the difficulty is how to find the largest ϵ such that $I(\hat{\theta}(\epsilon)) = m$. In §2, a continuation approach is used to implement the strategy for the selection of ϵ based on control of $I(\theta)$. Further discussions of (8) as a nonparametric smoothing method are also given in §2.

Section 3 is devoted to the related statistical properties of the piecewise convex/concave smoother.

Most nonparametric smoothing techniques studied in the literature are linear for any fixed smoothing parameter (cf. [19]), whereas the proposed smoother is nonlinear with respect to the raw data. However, some similarity with spline smoothing is discussed in §4. Some discrepancy in using the classical "leave-one-out" cross-validation scheme to determine the smoothing parameter for spline smoothing is pointed out. It is noted that for a spline version of the proposed smoother the classical cross-validation technique can be correctly adopted. The detailed analysis of this observation will be done later and reported elsewhere.

In §5 we consider eleven simulation models studied by previous authors and apply our smoothing procedure on them. These models are included to study the effect of the variance of noise, the location of x_i's, the sample size, the shape of the underlying smooth curves, and outliers on the smoother. Several performance parameters, which are important and essential in any simulation analysis of a model, are included in the study. Also, we repeat the numerical simulations done by Mammen [26] to show that (8) with the proposed strategy for the selection of ϵ is very similar to (3) with $k = 1$.

Some final conclusions and future research problems are presented in §6.

2. A piecewise convex/concave smoother. We first give a detailed description of the proposed data smoothing algorithm controlled by $I(\theta)$. Then a continuation approach is given to implement the computation of the piecewise convex/concave smoother.

In the introduction, for convenience, we identified a vector θ with its linear spline interpolator so that $I(\theta)$ and $M(\theta)$ can also be defined for vectors. However, for piecewise linear functions, $I(\theta)$ and $M(\theta)$ can be computed easily. Given a vector $z \in \mathbf{R}^n$, we say that z has m sign changes, if there exist indices $0 =: i_0 < i_1 < \cdots < i_m < i_{m+1} := n$ such that the following hold:

1. for any given s, z_j's are not all zeros for $i_s < j \leq i_{s+1}$;
2. $z_i \cdot z_j \leq 0$ for $i_{s-1} < i \leq i_s, i_s < j \leq i_{s+1}$, and $1 \leq s \leq m$.

Then one can verify that $I(\theta) = m$ if and only if $\Delta_2\theta$ has $(m - 1)$ sign changes and $M(\theta) = m$ if and only if $\Delta_1\theta$ has $(m - 1)$ sign changes. Therefore, for a given vector θ, it is very easy to compute $I(\theta)$ and $M(\theta)$, and our regression scheme based on control of $I(\theta)$ can be stated as follows.

A piecewise convex/concave smoothing scheme. Let $\hat{\theta}(\epsilon)$ denote the solution to the following least squares problem:

$$\min_{-\epsilon \leq \Delta_3\hat{\theta} \leq \epsilon} \sum_{i=1}^{n} (\theta_i - y_i)^2. \tag{9}$$

Given a positive integer m and a positive integer r,

1. set $\delta := \frac{x_n - x_1}{n}$, $j := 0$, and $\epsilon_0 := 0$;
2. compute $\hat{\theta}(\epsilon_0)$ and s_0, the number of sign changes of $\Delta_2\hat{\theta}(\epsilon_0)$;
3. reset $j := j + 1$;
4. set $\epsilon_j := \epsilon_{j-1} + \delta$, if $\epsilon < r\delta$; or $\epsilon_j := \epsilon_{j-1}\left(1 + \frac{1}{r}\right)$, if $\epsilon \geq r\delta$;
5. compute $\hat{\theta}(\epsilon_j)$ and s_j, the number of sign changes of $\Delta_2\hat{\theta}(\epsilon_j)$;
6. if $s_j \geq m$, then output the smoother $\bar{\theta} := \hat{\theta}(\epsilon_{j-1})$ and stop; otherwise, go back to step 3.

Remarks. 1. Note that the scalar δ is the average length of the subintervals $[x_i, x_{i+1}]$. Let $\theta_i = g(x_i)$ for $1 \leq i \leq n$. Then $(\Delta_3\theta)_i = \frac{g^{(3)}(\xi_i)}{6}$ for some ξ_i between x_i and x_{i+3}. The uncertainty of ξ_i produces a possible error of magnitude δ. Therefore, $\hat{\theta}(\epsilon)$ and $\hat{\theta}(\epsilon')$ are regarded as "indistinguishable", if $\|\epsilon - \epsilon'\| \leq \delta$. However, if we only use $\epsilon_j := \epsilon_{j-1} + \delta$ to increase the bound on the third divided difference, then it might take thousands of iterations in the above regression scheme to find the right value of ϵ. This is not practical. Therefore, we allow a relative error tolerance in ϵ of $\frac{100}{r}\%$; i.e., if $\frac{\|\epsilon-\epsilon'\|}{\|\epsilon\|} < \frac{100}{r}\%$, then $\hat{\theta}(\epsilon)$ and $\hat{\theta}(\epsilon')$ are considered as "indistinguishable." With $\epsilon_j := \epsilon_{j-1}\left(1 + \frac{1}{r}\right)$, ϵ_j grows exponentially as j increases. This enables the scheme to handle problems with a large optimal smoothing parameter $\hat{\epsilon}$ defined by

$$\hat{\epsilon} := \sup \left\{ \epsilon : \Delta_2 \hat{\theta}(\epsilon) \text{ has at most } (m-1) \text{ sign changes or } I(\hat{\theta}(\epsilon)) \leq m \right\}.$$

Notice that a larger r produces a more accurate choice of ϵ; while smaller r means less computational cost.

2. If possible, we want to compute $\hat{\theta}(\hat{\epsilon})$. If $I(\hat{\theta}(\epsilon))$ is a nondecreasing function of ϵ, then the above regression scheme does produce $\hat{\theta}(\hat{\epsilon})$ with reasonable error tolerance as explained in remark 1. To prevent the case in which we might miss a larger ϵ with $I(\hat{\theta}(\epsilon)) \leq m$, we actually computed a few more (up to 25) $\hat{\theta}(\epsilon_j)$'s after we found $\bar{\theta}$ in our preliminary numerical tests. However, all of those $I(\hat{\theta}(\epsilon_j))$'s are larger than $I(\bar{\theta})$. Therefore, we are confident about the above regression scheme even without the proof that $I(\hat{\theta}(\epsilon))$ is a nondecreasing function of ϵ.

3. Note that, if $\epsilon \geq \|\Delta_3 y\|_\infty =: \epsilon_\infty$, then $\hat{\theta}(\epsilon_\infty) \equiv y$. It seems a good idea to use bisection of the interval $[0, \epsilon_\infty]$ to find the optimal smoothing parameter $\hat{\epsilon}$, under the assumption that, in general, the number of sign changes of $\Delta_2 \hat{\theta}(\epsilon)$ is a nondecreasing function of ϵ. However, we can use the Lagrange multiplier of $\hat{\theta}(\epsilon_{j-1})$ as an initial guess of the Lagrange multiplier of $\hat{\theta}(\epsilon_j)$ to achieve a very efficient implementation of the above regression scheme. This is in essence a continuation method. In order to compute $\hat{\theta}(\epsilon_j)$, we start with the polynomial fitting $\hat{\theta}(0)$ which is extremely easy to compute, then gradually move to the solution $\hat{\theta}(\epsilon_j)$ as ϵ changes from 0 to ϵ_j.

4. When $\Delta_2 \hat{\theta}(\epsilon_j)$ has at least m sign changes and $\Delta_2 \hat{\theta}(\epsilon_{j-1})$ has at most $(m-2)$ sign changes, we know that neither $\hat{\theta}(\epsilon_{j-1})$ nor $\hat{\theta}(\epsilon_j)$ has the required structure and the smoothing process has failed. In this circumstance, under-smoothing seems to be more desirable than over-smoothing. Therefore, one might select $\hat{\theta}(\epsilon_j)$ as the smoother instead of $\hat{\theta}(\epsilon_{j-1})$.

5. In our preliminary numerical tests, we also tested (8) with different k's. For $k \geq 3$, the solutions are more or less the same, while the solutions are a little bit rougher if $k \leq 2$. Since a larger k increases the ill-conditioning of (8) [24], [5], we choose $k = 3$.

6. One might also use $M(\theta)$ to control the smoothing process. However, $I(\theta)$ provides an implicit control on $M(\theta)$, since $M(\theta) \leq I(\theta) + 1$. But one can construct θ such that $I(\theta)$ is large and $M(\theta) = 0$. Hence it is more effective to use $I(\theta)$ as a geometric shape parameter.

7. Let $K_h(x)$ be a kernel smoother of y with the standard normal density as the kernel function, and $\nu_k(g)$ denote the number of sign changes of the kth derivative of $g(x)$. Then Silverman [37] proved that $\nu_k(K_h)$ is a right continuous decreasing function of h and used the modes to control the kernel smoothing process in the same way as we use $I(\theta)$: for a given set of sample data $\{(x_i, y_i)\}$, find K_{h_n} where h_n is defined as

$$h_n := \inf \{h : K_h \text{ has at most } m \text{ modes}\}.$$

Silverman [37], [38] used h_n to test the null hypothesis that $f(x)$ has m modes against the alternative in which $f(x)$ has more than m modes. For density function estimation, Cuevas and Gonzalez-Manteiga [4] also proposed to use the minimum h such that K_h has m inflection points.

The algorithm for computing $\hat{\theta}(\epsilon_j)$ is essentially the same algorithm proposed in [24] for solving the k-convex approximation problem. In order to understand why the continuation method works here, we need a dual reformulation of (8) as an unconstrained minimization of a strictly convex quadratic spline function. First, one could verify that the Karush-Kuhn-Tucker condition of (8) is equivalent to the following system of piecewise linear equations:

$$\text{(10)} \qquad \theta = y + \Delta_k^T w, \; w = (Bw + c)_+ - (d - Bw)_+,$$

where $B := I - \alpha\Delta_k\Delta_k^T$, I is the $(n-k)\times(n-k)$ identity matrix, α is any positive scalar, $c, d \in \mathbf{R}^{n-k}$, with $c_i := \alpha(-\epsilon - (\Delta_k y)_i)$ and $d_i := \alpha((\Delta_k y)_i - \epsilon)$, w in $\mathbf{R}^{n-k}$ is the Lagrange multiplier, Δ_k^T denotes the transpose of the matrix Δ_k, and z_+ is the vector whose ith component is $\max\{z_i, 0\}$. Let $\varphi(w) := w - (Bw + c)_+ + (d - Bw)_+$. Then φ is a linear transformation of the gradient of the following quadratic spline function:

$$\Phi(w) := \frac{1}{2}w^T Bw - \frac{1}{2}\|(Bw + c)_+\|^2 - \frac{1}{2}\|(d - Bw)_+\|^2, \tag{11}$$

where $\|z\|^2 = \sum_{i=1}^{n-k}|z_i|^2$. The explicit relation is the following:

$$\Phi'(w) = B\varphi(w). \tag{12}$$

If $0 < \alpha < \|\Delta_k\Delta_k^T\|^{-1}$, where $\|\Delta_k\Delta_k^T\|$ is the spectral radius of $\Delta_k\Delta_k^T$, then B is nonsingular and Φ is a strictly convex spline function. Therefore, (8) is equivalent to the following unconstrained minimization problem:

$$\min_{w\in\mathbf{R}^{n-k}} \Phi(w). \tag{13}$$

Hence, if $\hat{w}$ is a solution of (13), then $\hat{\theta} := y + \Delta_k^T\hat{w}$ is the solution of (8). See [24] for the implementation of a Newton method for solving (13), which uses only finitely many flops (in exact arithmetic) to find the solution.

Now, if $|\epsilon_j - \epsilon_{j-1}|$ is very small, one can actually prove that the difference between the corresponding Lagrange multipliers (w's) is also small. Therefore, the Lagrange multiplier of $\hat{\theta}(\epsilon_{j-1})$ is an excellent initial guess of the Lagrange multiplier of $\hat{\theta}(\epsilon_j)$. This makes the above regression scheme computationally much easier than it appears to be.

3. Some statistical properties. It is not an uncommon practice in statistics to first reduce the noise in data using some smoothing techniques and then fit a model on smoothed data for further analysis (cf. [29]). Suppose the function $g(x)$ is a polynomial in x, of degree $(k-1)$, and it is contaminated by a very noisy error variable. Suppose we observe the data $y_i = f(x_i) + \eta_i$, $i = 1, \dots, n$. If we fit a polynomial model of degree $(k-1)$ in x for these data $y_1, \dots, y_n$, the solution is given by Py, where the matrix $P = X(X'X)^{-1}X'$, X being the matrix of polynomial terms of $x_1, \dots, x_n$; i.e.,

$$X = \begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{k-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{k-1} \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ 1 & x_n & x_n^2 & \cdots & x_n^{k-1} \end{pmatrix}.$$

Instead of this, noting the noisy appearance of the data, suppose we first smooth the data using a piecewise convex/concave smoother and then fit the polynomial model. It can be easily shown that the result is again Py. At least in this special case we see that no useful information is lost in the smoothing process.

Two important measures for the precision of an estimator are bias and mean squared error (MSE) of the estimator. In the present case, the bias (or the squared bias) of $\hat{f}$ is measured by

$$\|E(\hat{f}) - f\|^2 = \frac{1}{n}\sum_{i=1}^{n}[E(\hat{f}(x_i)) - f(x_i)]^2 \tag{14}$$

and the MSE is measured by

$$E\|\hat{f} - f\|^2 = E\left\{\frac{1}{n}\sum_{i=1}^{n}[\hat{f}(x_i) - f(x_i)]^2\right\}. \tag{15}$$

Again assume that $f(x)$ is a polynomial of degree $(k-1)$ in x. Then it can be shown that the bias (14) is zero. In order to prove this, we reformulate (8) to get a better understanding of the relationship between y and $\hat{f}$.

From the Karush-Kuhn-Tucker conditions we know that (8) is equivalent to the following problem:

$$\min \|\Delta_k^T w\|^2 \text{ subject to } -\epsilon \le \Delta_k y + \Delta_k \Delta_k^T w \le \epsilon, \tag{16}$$

in the sense that $\hat{f} = y + \Delta_k^T \hat{w}$ if and only if $\hat{w}$ is the solution of (16). Note $\Delta_k X = 0$, which implies $\Delta_k y = \Delta_k \eta$. Therefore, (16) can be rewritten as

$$\min \|\Delta_k^T w\|^2 \text{ subject to } -\epsilon \le \Delta_k \eta + \Delta_k \Delta_k^T w \le \epsilon. \tag{17}$$

Since (17) is invariant if we replace η by $-\eta$ and w by $-w$, the solution $\hat{w}(\equiv \hat{w}(\epsilon, \eta))$ of (16) is an odd mapping of η; i.e., $\hat{w}(\epsilon, -\eta) = -\hat{w}(\epsilon, \eta)$. Therefore, if the density function of η is an even function, then $E(\hat{w}) = 0$, which implies $E(\hat{f}) = E(y + \Delta_k \hat{w}) = E(y) + \Delta_k E(\hat{w}) = E(y)$.

Now let us assume that $y = f(x) + \eta$ as before and the kth derivative $f^{(k)}(x)$ of $f(x)$ exists. Then using Taylor's expansion we can write

$$\begin{aligned} f(x) &= f(\bar{x}) + f'(\bar{x})(x-\bar{x}) + \frac{f''(\bar{x})}{2!}(x-\bar{x})^2 \\ &\quad + \cdots + \frac{f^{(k-1)}(\bar{x})}{(k-1)!}(x-\bar{x})^{k-1} + \frac{f^{(k)}(t)}{k!}(x-\bar{x})^k, \end{aligned}$$

where $\bar{x} = \frac{1}{n}\sum_{i=1}^n x_i$ and t is some point between x and $\bar{x}$. This can be rewritten as

$$f(x_i) = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_{k-1} x_i^{k-1} + \beta_{ki} x_i^k,$$

where $\beta_{ki} = \frac{f^{(k)}(t_i)}{k!}$ and t_i is between x_i and $\bar{x}$ for $i = 1, \dots, n$.

Thus we have $y_i = f(x_i) + \eta_i$ or $y = X\beta + d + \eta$, where $\beta' = (\beta_0, \dots, \beta_{k-1})$, $d' = (d_1, \dots, d_n)$, and $d_i = \beta_{ki} x_i^k$.

If we set $u = y - d$, then we have $u = X\beta + \eta$, which is the polynomial model considered as before, and for this we know $E(u) = E(\hat{f})$. Thus, bias (14) is $\frac{1}{n}\sum_{i=1}^n \beta_{ki}^2 x_i^{2k}$ which is in terms of the kth derivative of the function $f(x)$.

Computation of bias in general is not easy. It involves the expected value of a certain truncated distribution. Even in the simplest of cases, computation of MSE would be difficult. We consider the following simple example which indicates how the bias and the variance (MSE-Bias^2) of the estimator depend on a certain truncated distribution.

Let $y_1 := c_1 + \eta_1$ and $y_2 := c_2 + \eta_2$, where c_1 and c_2 are real numbers and η_1 and η_2 are two independent $N(0, 1)$ random variables. Then (8) reduces to the following simple problem:

$$\min \frac{1}{2}(y_1 - \theta_1)^2 + \frac{1}{2}(y_2 - \theta_2)^2 \text{ subject to } -\epsilon \le \theta_2 - \theta_1 \le \epsilon. \tag{18}$$

It is easy to verify that the solution $\hat{\theta}$ should satisfy the following conditions: if $\epsilon = 0$, then $\hat{\theta}_1 = \hat{\theta}_2 = \frac{y_1+y_2}{2}$; if $|y_2 - y_1| \le \epsilon$, then $\hat{\theta}_1 = y_1$ and $\hat{\theta}_2 = y_2$; if $y_2 - y_1 > \epsilon$, then $\hat{\theta}_1 = \frac{y_1+y_2}{2} - \frac{\epsilon}{2}$ and $\hat{\theta}_2 = \frac{y_1+y_2}{2} + \frac{\epsilon}{2}$; if $y_2 - y_1 < -\epsilon$, then $\hat{\theta}_1 = \frac{y_1+y_2}{2} + \frac{\epsilon}{2}$ and $\hat{\theta}_2 = \frac{y_1+y_2}{2} - \frac{\epsilon}{2}$. If we denote the solution of (18) by $(\hat{\theta}_1(\epsilon), \hat{\theta}_2(\epsilon))$, then

$$\hat{\theta}_1(\epsilon) = \begin{cases} \frac{y_1+y_2}{2} + \frac{\epsilon}{2}, & \text{if } y_2 - y_1 < -\epsilon, \\ \frac{y_1+y_2}{2} + \frac{y_1-y_2}{2}, & \text{if } |y_2 - y_1| \le \epsilon, \\ \frac{y_1+y_2}{2} - \frac{\epsilon}{2}, & \text{if } y_2 - y_1 > \epsilon, \end{cases}$$

and

$$\hat{\theta}_2(\epsilon) = \begin{cases} \frac{y_1+y_2}{2} - \frac{\epsilon}{2}, & \text{if } y_2 - y_1 < -\epsilon, \\ \frac{y_1+y_2}{2} + \frac{y_2-y_1}{2}, & \text{if } |y_2 - y_1| \leq \epsilon, \\ \frac{y_1+y_2}{2} + \frac{\epsilon}{2}, & \text{if } y_2 - y_1 > \epsilon. \end{cases}$$

Now we want to compute the mean and the variance of $\hat{\theta}_i(\epsilon)$, $i = 1, 2$, and also the covariance between $\hat{\theta}_1(\epsilon)$ and $\hat{\theta}_2(\epsilon)$. For this, write $\hat{\theta}_1(\epsilon) = u + v$ and $\hat{\theta}_2(\epsilon) = u - v$, where $u := \frac{y_1+y_2}{2}$ and

$$v = \begin{cases} \frac{\epsilon}{2}, & \text{if } y_2 - y_1 < \epsilon, \\ \frac{y_1-y_2}{2}, & \text{if } -\epsilon \leq y_2 - y_1 \leq \epsilon, \\ -\frac{\epsilon}{2}, & \text{if } y_2 - y_1 > \epsilon. \end{cases}$$

We observe that $E(\hat{\theta}_1(\epsilon)) = E(u) + E(v)$, $E(\hat{\theta}_2(\epsilon)) = E(u) - E(v)$, $\text{var}(\hat{\theta}_1(\epsilon)) = \text{var}(\hat{\theta}_2(\epsilon)) = \text{var}(u) + \text{var}(v)$, and $\text{cov}(\hat{\theta}_1(\epsilon), \hat{\theta}_2(\epsilon)) = \text{var}(u) - \text{var}(v)$. Furthermore, $E(u) = \frac{c_1+c_2}{2}$, $\text{var}(u) = \frac{1}{2}$, and

$$\begin{aligned} E(v) &= E[vI_{\{y_2-y_1<-\epsilon\}}] + E[vI_{\{-\epsilon\leq y_2-y_1\leq\epsilon\}}] + E[vI_{\{y_2-y_1>\epsilon\}}] \\ &= \tfrac{\epsilon}{2}\Pr[y_2 - y_1 < -\epsilon] - \tfrac{1}{2}E[(y_2 - y_1)I_{\{-\epsilon\leq y_2-y_1\leq\epsilon\}}] - \tfrac{\epsilon}{2}\Pr[y_2 - y_1 > \epsilon], \end{aligned}$$

where I_A is the indicator of the event A; i.e., $I_A = 1$ if A is true and 0 otherwise. Notice that the middle term is the expected value of the truncated normal distribution with mean $c_2 - c_1$, variance two, and truncated at $-\epsilon$ and ϵ. (For a discussion of the truncated normal distribution, see [23].) Therefore, we can write

$$E(v) = \frac{\epsilon}{2}\Phi(t_1) - \frac{c_2 - c_1}{2} - \frac{\sqrt{2}}{2} \cdot \frac{\phi(t_1) - \phi(t_2)}{\Phi(t_2) - \Phi(t_1)} - \frac{\epsilon}{2}(1 - \Phi(t_2)),$$

where

$$\phi(t) := \frac{1}{\sqrt{2\pi}}e^{-t^2/2}, \quad \Phi(t) := \int_{-\infty}^{t} \phi(s)ds,$$

$$t_1 := \frac{-\epsilon - (c_2 - c_1)}{\sqrt{2}} \quad \text{and} \quad t_2 := \frac{\epsilon - (c_2 - c_1)}{\sqrt{2}}.$$

Similar calculations show that

$$\begin{aligned} \text{var}(v) = \tfrac{1}{2}&\left\{1 + \tfrac{t_1\phi(t_1)-t_2\phi(t_2)}{\Phi(t_2)-\Phi(t_1)} - \left(\tfrac{\phi(t_1)-\phi(t_2)}{\Phi(t_2)-\Phi(t_1)}\right)^2\right\} \\ &+\tfrac{\epsilon^2}{4}\Phi(t_1) + \tfrac{\epsilon^2}{4}[1 - \Phi(t_2)]. \end{aligned}$$

In the special case when $c_1 = c_2$, we see that $E(v) = 0$ and

$$\text{var}(v) = \frac{\epsilon^2}{2}\Phi(-\epsilon/\sqrt{2}) + \frac{1}{2} \cdot \left(1 - \frac{\epsilon\phi(\epsilon)}{\Phi(\epsilon) - 0.5}\right).$$

Since the first term in $\text{var}(v)$ tends to zero as $\epsilon \to 0$ as well as $\epsilon \to \infty$,

$$\lim_{\epsilon\to 0+} \frac{\epsilon\phi(\epsilon)}{\Phi(\epsilon) - 0.5} = 1, \quad \text{and} \quad \lim_{\epsilon\to\infty} \frac{\epsilon\phi(\epsilon)}{\Phi(\epsilon) - 0.5} = 0,$$

we obtain

$$\lim_{\epsilon \to 0} \operatorname{var}(v) = 0 \quad \text{and} \quad \lim_{\epsilon \to \infty} \operatorname{var}(v) = \frac{1}{2}.$$

The above calculations show that $\hat{\theta}_i(\epsilon)$'s are unbiased estimators of θ_i's only when

$$\frac{\epsilon}{2}\Phi(t_1) - \frac{\sqrt{2}}{2} \cdot \frac{\phi(t_1) - \phi(t_2)}{\Phi(t_2) - \Phi(t_1)} - \frac{\epsilon}{2}(1 - \Phi(t_2)) = 0,$$

i.e., when $c_1 = c_2$.

This simple example indicates how difficult it is, in general, to compute the bias and the variance of the estimator. Therefore, in the numerical examples that we consider later in §5, we use Monte-Carlo simulation to compute the bias and the MSE. That is, we compute $\mathbf{B}_S := \frac{1}{n}\sum_{i=1}^n |E\hat{f}(x_i) - f(x_i)|^2$ and $\mathbf{E}_S := \frac{1}{n}E\left(\sum_{i=1}^n |\hat{f}(x_i) - f(x_i)|^2\right)$, where the expected value, $E(\cdot)$, is the average over 1,000 simulations.

4. Comparison with spline smoothing. Although we treat our piecewise convex/concave smoother as a shape preserving estimator of the underlying function $f(x)$, (8) can also be viewed as a nonparametric regression method with ϵ as the smoothing parameter. Nonparametric smoothing is very useful and flexible for exploring a general relationship between two variables or a general pattern in data. Every nonparametric model is tuned by some smoothing parameter (such as ϵ in (8)) which balances the degree of fidelity to the data against the smoothness of the estimated curve (cf. [19]). Most nonparametric smoothing techniques, such as moving averaging, k-nearest neighbor estimates, kernel smoothing, orthogonal series estimates, spline smoothing, etc., are linear in the sense that the estimator is a linear mapping of the raw data for any fixed smoothing parameter (cf. [19]) except median smoothing and M-smoothing. As we mentioned before, the proposed data smoother is nonlinear with respect to the raw data. However, despite its nonlinearity, it is very similar to spline smoothing models. In this section, we try to analyze the similarity and the difference between (8) and spline smoothing.

Let $S_{n,k}$ be the space of all splines of order $(k+1)$ and with knot sequence $x_1 < \cdots < x_n$ on $[a, b]$; i.e., $g \in S_{n,k}$ if and only if $g(x)$ is a polynomial of degree at most k on each subinterval $[x_{i-1}, x_i]$ for $2 \le i \le n$, and the $(k-1)$th derivative $g^{(k-1)}(x)$ of $g(x)$ is continuous. The spline smoothing model can be formulated as follows:

$$\min_{g \in S_{n,k}} \sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int_a^b |g^{(k)}(x)|^2 dx, \tag{19}$$

where λ is the smoothing parameter. The solution of (19) is called the smoothing spline of Schoenberg and Reinsch by de Boor (cf. pp. 235-237 in [2]), due to the fact that Schoenberg [36] and Reinsch [33] were the first to study such a smoothing technique. However, the idea goes back to Whittaker [42], who proposed the following data smoothing scheme:

$$\min \sum_{i=1}^{n} (y_i - \theta_i)^2 + \lambda \sum_{i=1}^{n-k} (\Delta_k \theta_i)^2. \tag{20}$$

First, note that the kth divided difference in (20) is replaced by the kth derivative in (19) and that the summation in (20) is replaced by the integration in (19). Therefore, (19) can be considered as the spline version of (20).

It is well known that, for each $\lambda \geq 0$, there is a unique $\epsilon \geq 0$ such that (19) is equivalent to the following constrained optimization problem:

$$\begin{array}{l} \min \sum_{i=1}^{n}(y_i - g(x_i))^2 \\ \text{subject to } g \in S_{n,k} \text{ and } \int_a^b |g^{(k)}(x)|^2 dx \leq \epsilon. \end{array} \tag{21}$$

Note that a smaller ϵ in (21) (corresponding to a larger λ in spline) yields a smoother spline fitting of y. While ϵ provides explicit control on the magnitude of the 2-norm of $g^{(k)}(x)$, (19) is much easier to solve than (21).

Similarly, if we define $\|y\|_\infty := \max_{1 \leq i \leq n} |y_i|$, then the penalty function formulation of (8) yields the following unconstrained minimization problem:

$$\min \sum_{i=1}^{n}(y_i - \theta_i)^2 + \lambda \|\Delta_k \theta\|_\infty, \tag{22}$$

where λ is some nonnegative number. Notice the similarity between (22) and (20). In essence, (22) is (20) with the l_∞-norm replacing the l_2-norm. Hence, (8) can be viewed as the discretized version of (19) with the l_2-norm being replaced by the l_∞-norm. As a consequence of this observation, the spline version of (8) should be the following:

$$\begin{array}{l} \min \sum_{i=1}^{n}(y_i - g(x_i))^2 \\ \text{subject to } g \in S_{n,k} \text{ and } -\epsilon \leq g^{(k)}(x) \leq \epsilon \text{ for } x \neq x_i,\ 1 \leq i \leq n. \end{array} \tag{23}$$

Recall that $g(x)$ is a polynomial of degree at most k on each subinterval $[x_{i-1}, x_i]$. Thus, $g^{(k)}(x)$ is constant for $x \in [x_{i-1}, x_i]$ for a fixed i. Therefore, (23) is actually a convex quadratic programming problem with $(n-1)$ linear constraints. Similarly, (23) can be reformulated as

$$\min_{g \in S_{n,k}} \sum_{i=1}^{n}(y_i - g(x_i))^2 + \lambda \max_{a \leq x \leq b} |g^{(k)}(x)|. \tag{24}$$

Since the constraint in (21) is a differentiable mapping of g, the penalty function approach yields a simple smooth unconstrained minimization problem which is very easy to solve. However, the penalty function approach applied to (8) or (23) transforms a linearly constrained quadratic programming problem into a nonsmooth unconstrained minimization problem (22) or (24), which seems more difficult to solve than (8) or (23).

Our numerical experiments indicate that it takes about $\mathcal{O}(n^2)$ flops to solve (8) (cf. [24] and Table 1), which is comparable with the computational cost of solving (19). We believe that the computational cost of solving (23) is more or less $\mathcal{O}(n^3)$ flops, due to a full constraint matrix. One difference between (8) and (19) is that ϵ directly controls the magnitude of the kth difference of the fitting curve in (8), while λ indirectly controls the magnitude of the kth derivative of the fitting curve in (19). Practically, they both achieve the purpose of controlling the smoothness of fitting curves. For example, the shape control strategy in the previous section can be modified to determine the smoothing parameter λ in (19). However, there is some fundamental difference in statistical analysis of a constrained data fitting model and its penalty function reformulation.

Suppose we want to fit data by a model $F(\alpha)$ involving the parameter α; i.e., for a fixed α, $F(\alpha)$ is a set of fitting curves. Then the classical "leave-one-out" cross-validation scheme is a means to determine which α provides the best model for the given data.

Consider the "leave-one-out" cross-validation scheme for the determination of λ in (19) and ϵ in (21):

$$\min_{g \in S_{n,k}} \sum_{i=1, i \neq j}^{n} (y_i - g(x_i))^2 + \lambda \int_a^b |g^{(k)}(x)|^2 dx \tag{25}$$

TABLE 1
Numerical simulation results.

Model	m	$\mathbf{E_0}$	$\mathbf{E_S}$	$\mathbf{B_S}$	ϵ_S	$\mathbf{R_S}$	$\mathbf{E_I}$	$\mathbf{B_I}$	ϵ_I	$\mathbf{R_I}$	**N**	$\mathbf{N_0}$	**CPU**
1	8	0.40E−01	0.47E−02	0.93E−03	0.38E+01	88%	0.40E−02	0.41E−03	0.50E+01	89%	409	4.6	2.3
2	1	0.10E+01	0.72E−01	0.86E−05	0.65E+01	92%	0.61E−01	0.60E−05	0.00E+00	93%	167	2.1	0.4
3	3	0.23E−05	0.22E−06	0.87E−09	0.61E+00	90%	0.16E−06	0.17E−07	0.35E+00	92%	167	3.8	0.7
4	3	0.34E+00	0.33E−01	0.92E−02	0.78E+02	90%	0.22E−01	0.41E−02	0.94E+02	93%	406	2.9	2.0
5	4	0.10E−01	0.14E−02	0.81E−05	0.45E+03	86%	0.95E−03	0.12E−03	0.27E+03	90%	573	3.5	3.0
6	1	0.91E+01	0.31E+00	0.13E−02	0.14E+02	96%	0.27E+00	0.14E−02	0.27E+00	97%	250	2.2	1.2
7	8	0.10E+01	0.51E−01	0.98E−02	0.34E+03	94%	0.49E−01	0.20E−01	0.16E+03	95%	728	4.2	8.2
8	9	0.10E−01	0.11E−02	0.58E−03	0.12E+03	88%	0.85E−03	0.18E−03	0.31E+03	91%	601	4.5	6.7
9	9	0.25E+00	0.12E−01	0.60E−03	0.25E+03	95%	0.92E−02	0.20E−02	0.76E+02	96%	685	4.4	7.7
10	1	0.10E−01	0.39E−03	0.38E−05	0.32E+02	96%	0.22E−03	0.35E−04	0.15E+02	97%	427	3.7	4.5
11	1	0.25E+00	0.79E−02	0.20E−03	0.27E+02	96%	0.49E−02	0.58E−03	0.15E+02	98%	385	3.2	4.0

and

$$(26)\qquad \begin{array}{l} \min \sum_{i=1, i\neq j}^{n} (y_i - g(x_i))^2 \\ \text{subject to } g \in S_{n,k} \text{ and } \int_a^b |g^{(k)}(x)|^2 dx \leq \epsilon. \end{array}$$

Notice that, for the same ϵ, the equivalent penalty function reformulation of (26) has different λ values for different missing indices j. Therefore, the "leave-one-out" scheme has different meanings depending on whether it is applied to (19) or (21). From the data smoothing point of view, it seems easier to explain the statistical meaning of applying the "leave-one-out" scheme to (21). Since the fitting curve belongs to a class of spline functions whose 2-norms are bounded by $\sqrt{\epsilon}$, the scheme is to determine the ability of this class to predict a missing value. However, when applying the "leave-one-out" scheme to (19), the 2-norms of the fitting curves produced by (25) depend on the missing index j; i.e., for different indices, the fitting curves produced by (25) have different "degrees of smoothness." If one tries to explain the "leave-one-out" scheme to spline from the classical point of view, it will be difficult to figure out which parametric family of models is involved. Therefore, there is a fundamental difference between applying the "leave-one-out" scheme to (19) and to (21). However, the generalized cross-validation derived from applying the "leave-one-out" scheme to (19) is an effective way of selecting the smoothing parameter in (19) (cf. [40] and the references therein). Also, it is very interesting to see that, for a fixed λ, the smoother defined by (19) is a linear mapping of y, while the solution of (21) is a nonlinear mapping of y for a fixed ϵ.

5. Numerical examples. In this section, we discuss the numerical performance of the proposed regression scheme. In order to understand how the smoother is affected by different aspects of the contaminated data y, such as the variance of noise, the location of x_i's, the sample size n, the shape of the underlying smooth curve, and outliers, we select the following eleven simulation models considered earlier in the literature to test the proposed regression scheme. For each simulation model, we have performed 1000 simulations.

Numerical simulation models. Assume that η_i's are independent normal random variables with mean 0 and variance 1. We use $x \sim U(0, 1)$ to denote that x is a random variable which has the uniform distribution in the interval (0,1).

1. Wahba and Wold's simulation model (cf. [40] or [41]): $n = 100, \sigma = 0.2$,

$$f(x) := 4.26(e^{-x} - 4e^{-2x} + 3e^{-3x}),$$
$$x_i = \tfrac{3.25i}{n-1} \text{ and } y_i = f(x_i) + \sigma\eta_i, \; 1 \leq i \leq n.$$

2. Cleveland's simulation model [3]: $n = 50, \sigma = 1$,

$$f(x) := x,$$
$$x_i = \tfrac{i}{n} \text{ and } y_i = f(x_i) + \sigma\eta_i, \; 1 \leq i \leq n.$$

3. Rice's simulation model [34]: $n = 75, \sigma = 0.0015$,

$$f(x) := x^3(1-x)^3,$$
$$x_i = \tfrac{i-1}{n} \text{ and } y_i = f(x_i) + \sigma\eta_i, \; 1 \leq i \leq n.$$

4. Friedman's simulation model (cf. [15] or [19]): $n = 100, \sigma = 1$,

$$f(x) := \sin\left(2\pi(1-x)^2\right),$$
$$x_i \sim U(0,1) \text{ and } y_i = f(x_i) + \sigma x_i\eta_i, \; 1 \leq i \leq n.$$

5. Härdle and Bowman's simulation model [20]: $n = 100, \sigma = 0.1$,

$$f(x) := \sin(4\pi x),$$
$$x_i = \tfrac{i-\frac{1}{2}}{n} \text{ and } y_i = f(x_i) + \sigma\eta_i, \; 1 \leq i \leq n.$$

6. Härdle's simulation model I (cf. p. 77 of [19]): $n = 200, \sigma = 1$,

$$f(x) := 1 - x + e^{-200(x-\frac{1}{2})^2},$$
$$x_i \sim U(0,1) \text{ and } y_i = f(x_i) + \sigma\eta_i, \; 1 \leq i \leq n.$$

7. Härdle's simulation model II [18]: $n = 100, \sigma = 1$,

$$f(x) := \sin(\pi x),$$
$$x_i \sim U(0,1) \text{ and } t_i \sim U(0,1),$$
$$y_i = \begin{cases} f(x_i) + \sigma\eta_i, & \text{if } t_i > 0.1 \\ f(x_i) + 9\sigma\eta_i, & \text{if } t_i \leq 0.1, \end{cases} \quad 1 \leq i \leq n.$$

8. Mammen's simulation model I-1 [26]: $n = 200$, $\sigma = 0.1$, $f(x)$ is the broken line joining the points (0,0),(0.3,−1),(0.7,1), and (1,0),

$$x_i = \frac{i}{n} \text{ and } y_i = f(x_i) + \sigma\eta_i, \; 1 \leq i \leq n.$$

9. Mammen's simulation model I-2 [26]: the same as Mammen's simulation model I-1, except $\sigma = 0.5$.
10. Mammen's simulation model II-1 [26]: $n = 200, \sigma = 0.1$,

$$f(x) = 15x(x-0.5)(1-x),$$
$$x_i = \tfrac{i}{n} \text{ and } y_i = f(x_i) + \sigma\eta_i, \; 1 \leq i \leq n.$$

11. Mammen's simulation model II-2 [26]: the same as Mammen's simulation model II-1, except $\sigma = 0.5$.

Figures 1-7 correspond to the simulation models 1-7. The contaminated data $\{(x_i, y_i)\}_1^n$ are plotted as little circles, the smoother $\{(x_i, \hat{\theta}(\epsilon_1)_i)\}_1^n$ is plotted as the solid line, and the original curve $\{(x_i, f(x_i))\}_1^n$ is plotted as the dashed line.

Wahba and Wold's simulation model 1 is for comparison between our data smoothing model and the spline smoothing model. It is interesting to note that our smoother is almost identical to the smoother produced by Wahba and Wold (cf. Fig. 1).

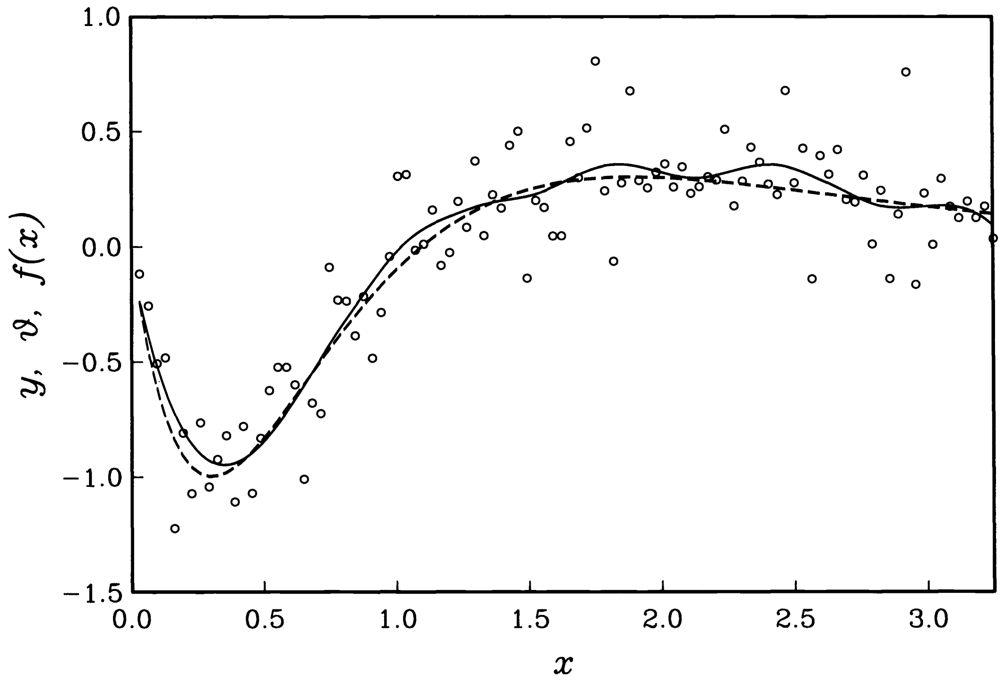


FIG. 1. *Simulation for* $f(x) = 4.26(e^{-x} - 4e^{-2x} + 3e^{-3x})$.

Simulation models 3 and 5 are for recovery of smooth regression functions without a "fairly flat" segment and we see that our procedure produces curves almost identical to the original curves (cf. Figs. 3 and 5).

Simulation models 2 and 7 are considered to study the effect of outliers or very noisy data (large variance relative to the signal) on the procedure. Simulation model 2 was used by Cleveland to illustrate locally weighted scatter plot smoothing (LOWESS) proposed by him. Our procedure identifies the underlying function fairly closely (cf. Fig. 2). Simulation model 7 was considered by Härdle to study the behavior of a kernel M-smoother on the data with outliers. Application of our procedure to this data indicates that the procedure is quite resistant to the presence of outliers. We want to point out that about 10% of the observations in this data set are extremely large in absolute magnitude as compared to the rest of the observations. However, our procedure is able to identify the true function hidden in this highly scattered cloud of data (cf. Fig. 7).

Simulation model 6 represents a somewhat extreme case where most of the signal is buried in the noise. Härdle contends that these types of data are not uncommon in practice. Even in this case our method is able to pick up the signal which is fairly close to the original curve (cf. Fig. 6).

Simulation model 4 was considered by Friedman to illustrate the behavior of a smoother introduced by him called supersmoother. Figure 4 indicates that our smoother picks up the original function well and is not sensitive to the location of x_i's.

Simulation models 8-11 are used to compare our piecewise convex/concave smoothing method with Mammen's piecewise convex/concave smoothing method. The mean squared errors, produced by our piecewise convex/concave estimator and by Mammen's piecewise convex/concave estimator, are comparable; this indicates that the proposed regression scheme is very similar to (3) with $k = 1$ and $D = \infty$.

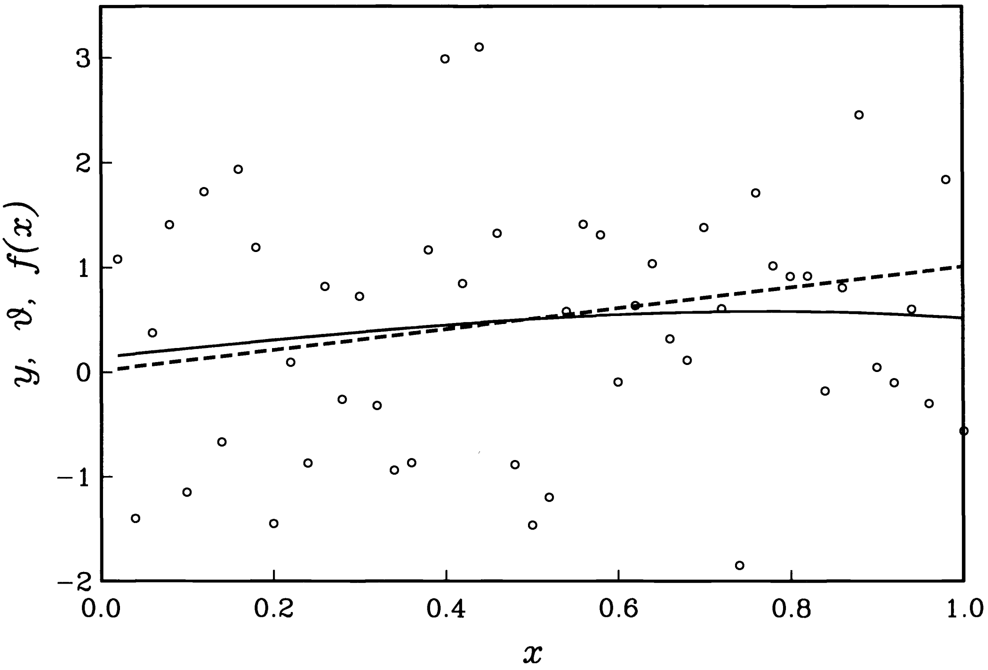


FIG. 2. *Simulation for* $f(x) = x$.

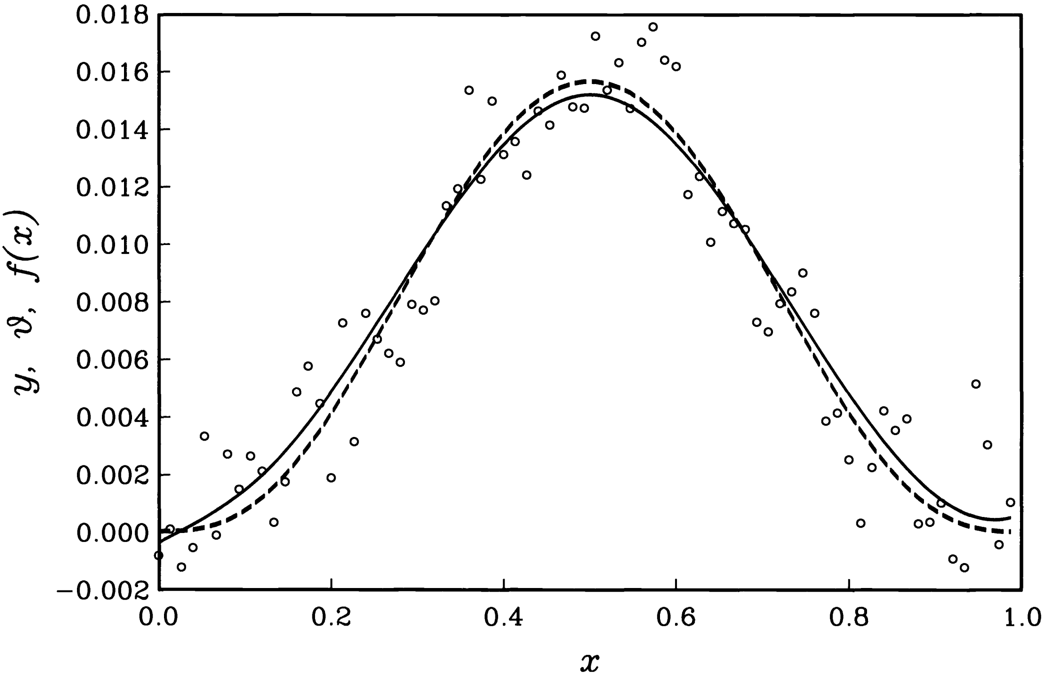


FIG. 3. *Simulation for* $f(x) = x^3(1-x)^3$.

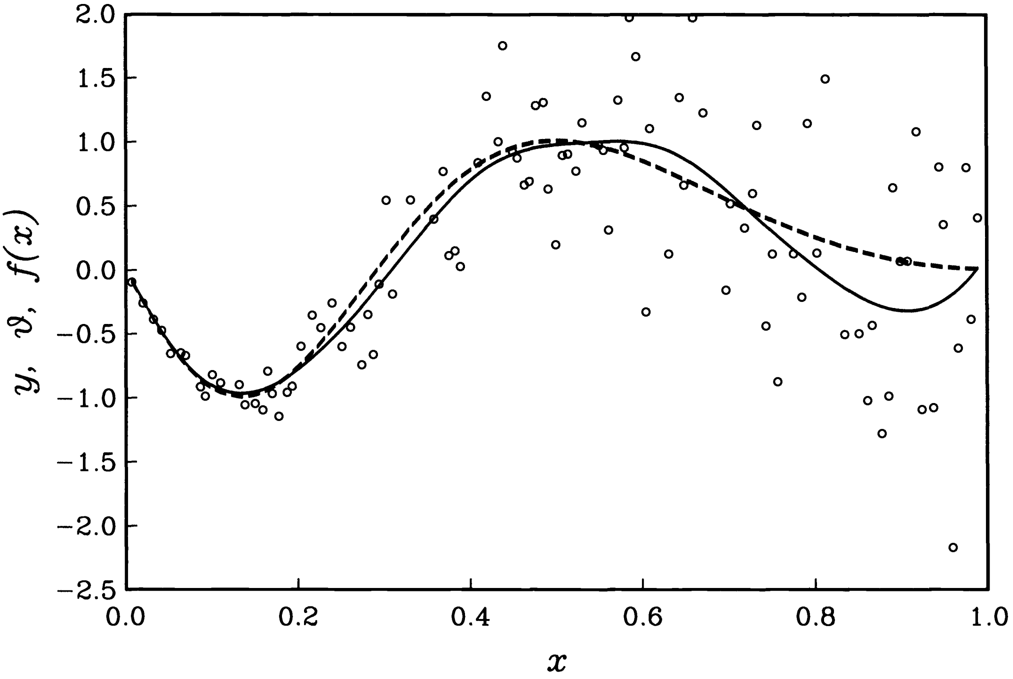


FIG. 4. *Simulation for* $f(x) = \sin(2\pi(1-x)^2)$.

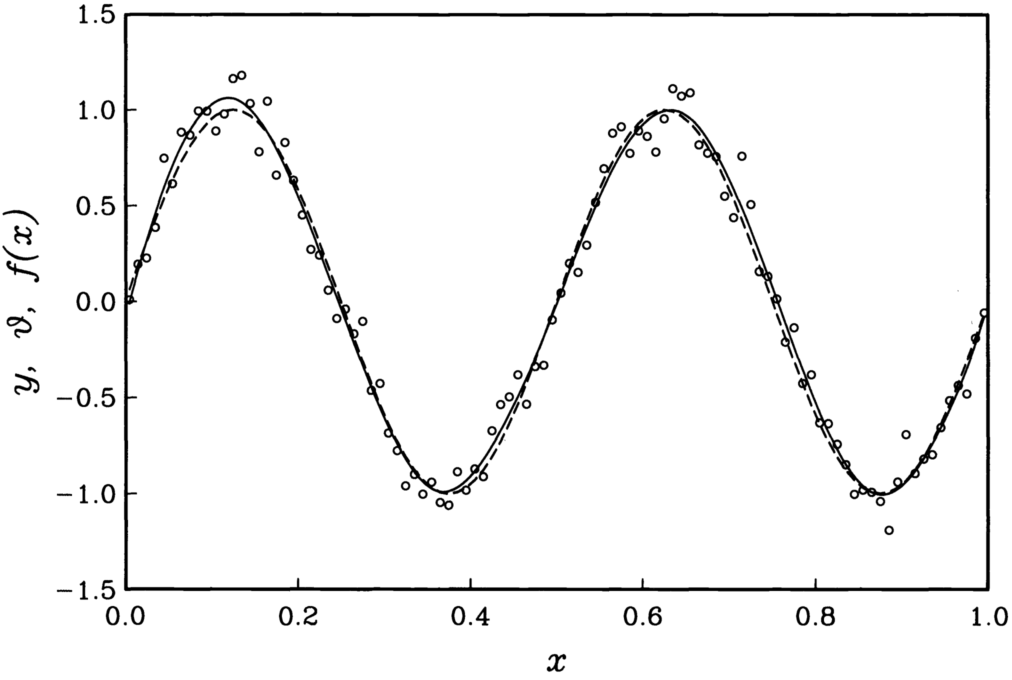


FIG. 5. *Simulation for* $f(x) = \sin(4\pi x)$.

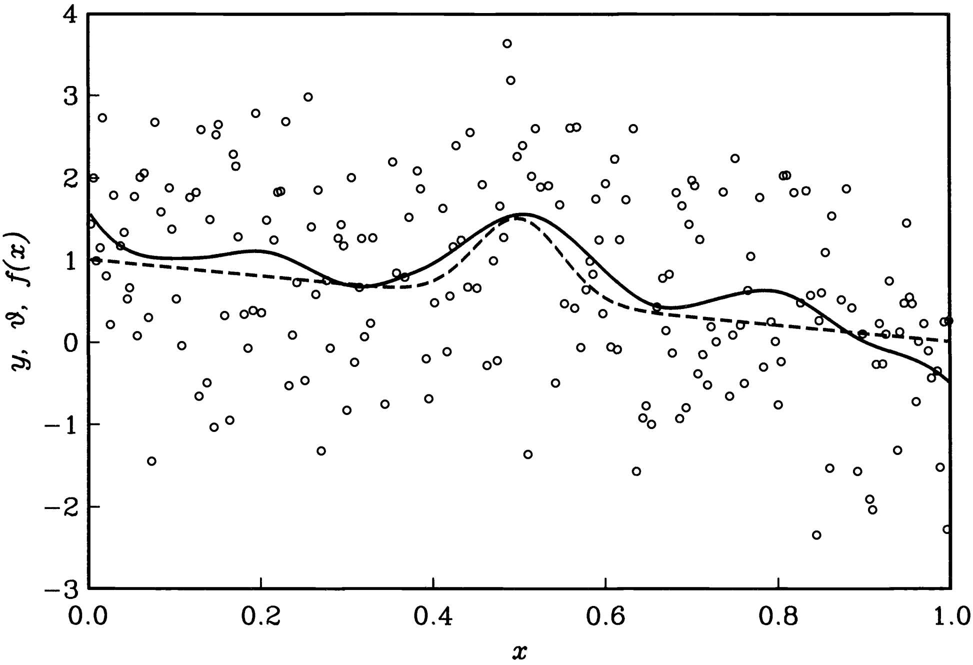


FIG. 6. *Simulation for* $f(x) = 1 - x + e^{-200(x-0.5)^2}$.

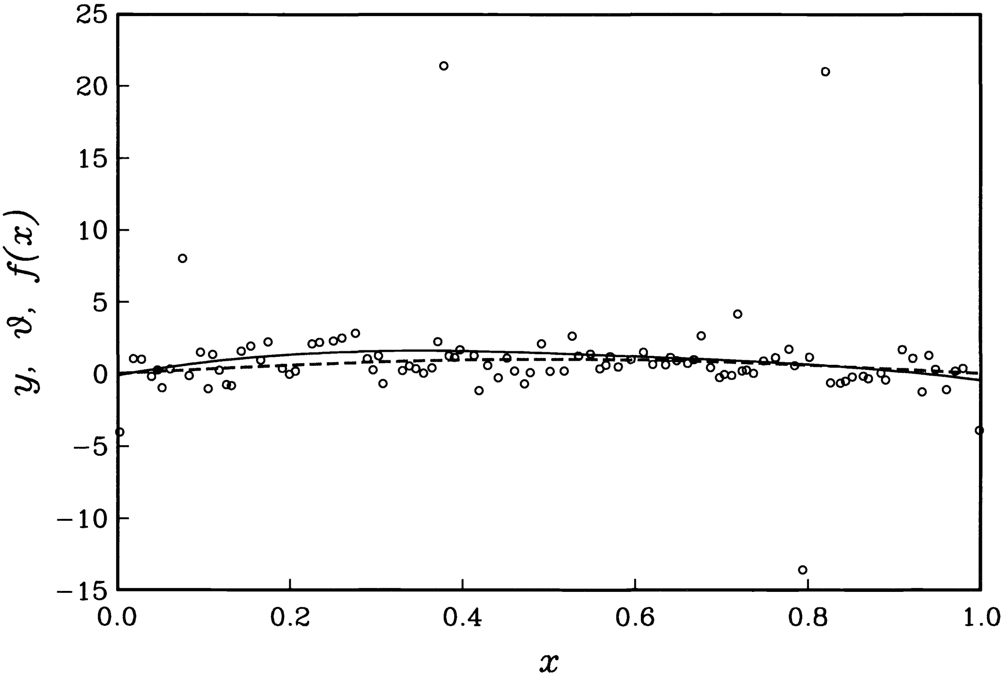


FIG. 7. *Simulation for* $f(x) = \sin(\pi x)$.

The only parameter needed for the regression scheme is the number of sign changes. When the structure of $f(x)$ is visible in $\{(x_i, y_i)\}_1^n$ (such as Rice's simulation model 3 and the Härdle-Bowman model 5), it is not difficult to find the right choice of m. However, in general, the parameter m is a means for exploring the structure of the underlying curve $f(x)$. In each of the above simulation models, we select the integer m such that the regression scheme produces $\hat{\theta}(\epsilon_{j-1})$, which is closest to $\{f(x_i)\}_1^n$. Our purpose is to show that the regression scheme does have some kind of ability to recover the original curve with an appropriate choice of the number of sign changes of the second divided difference of the fitting vector. For each model, we determine the best choice of m by experimenting on a set of simulation data and then use the same m for all 1000 simulations. The simulations indicate that, for a particular function $f(x)$, there is a corresponding optimal choice of m which will produce a curve closest to $f(x)$. The most interesting part is that, with appropriate choice of m, the corresponding smoother $\hat{\theta}(\epsilon_S)$ is more or less the same as the ideal smoother $\hat{\theta}(\epsilon_I)$, which is the solution closest to $f(x)$ among all $\theta(\epsilon)$'s. As a consequence, we can conclude that one can find the best fitting of $f(x)$ among all $\theta(\epsilon)$'s by selecting an appropriate m instead of ϵ. This is a very important property of our smoothing process, since the range of m is discrete and the range of ϵ is continuous. In a practical situation, it is easy to experiment with a few different values of m to decide which m produces the most desirable estimator.

There are several key performance parameters we want to measure when the regression scheme is applied to produce a smoother:

1. $\mathbf{E}_0 := \frac{1}{n}E\left(\sum_{i=1}^n |y_i - f(x_i)|^2\right)$, the mean squared error between f and the contaminated data y;
2. ϵ_S : the smoothing parameter determined by the regression scheme;
3. $\mathbf{E}_S := \frac{1}{n}E\left(\sum_{i=1}^n |\hat{\theta}(\epsilon_S)_i - f(x_i)|^2\right)$, the mean squared error between $f(x)$ and the smoother $\hat{\theta}(\epsilon_S)$;
4. $\mathbf{B}_S := \frac{1}{n}\sum_{i=1}^n |E\hat{\theta}(\epsilon_S)_i - f(x_i)|^2$, the squared bias of the smoother $\hat{\theta}(\epsilon_S)$;
5. $\mathbf{R}_S := (\mathbf{E}_S - \mathbf{E}_0)/\mathbf{E}_0$, the rate of noise reduction by the smoother $\hat{\theta}(\epsilon_S)$;
6. ϵ_I : the ideal smoothing parameter;
7. $\mathbf{E}_I := \frac{1}{n}E\left(\sum_{i=1}^n |\hat{\theta}(\epsilon_I)_i - f(x_i)|^2\right)$, the mean squared error between f and the ideal smoother $\hat{\theta}(\epsilon_I)$;
8. $\mathbf{B}_I := \frac{1}{n}E\sum_{i=1}^n |E\hat{\theta}(\epsilon_I)_i - f(x_i)|^2$, the squared bias of the ideal smoother $\hat{\theta}(\epsilon_I)$;
9. $\mathbf{R}_I := (\mathbf{E}_I - \mathbf{E}_0)/\mathbf{E}_0$, the rate of noise reduction by the ideal smoother $\hat{\theta}(\epsilon_I)$;
10. **N** : the number of Newton iterations involved in the algorithm (for solving (13)) which is used in step 5 of the piecewise convex/concave smoothing scheme;
11. $\mathbf{N}_0$: the average number of iterations involved in a line search for each Newton iteration;
12. **CPU**: the average CPU time (in seconds) for solving a simulation problem;

where the expected value, $E(\cdot)$, is the average over 1000 Monte-Carlo simulations, and ϵ_I is the so-called ideal smoothing parameter in the sense that

$$\sum_{i=1}^n |\hat{\theta}(\epsilon)_i - f(x_i)|^2$$

is minimum when $\epsilon = \epsilon_I$. The parameter $\mathbf{R}_I$ is a measurement of the capability of (9) as a data smoothing model, while the comparison of $\mathbf{R}_S$ and $\mathbf{R}_I$ provides a sense of intelligence of the regression scheme in determining the optimal smoothing parameter. The computational cost is about $\mathcal{O}(n\mathbf{N}_0\mathbf{N})$ flops [24], while **CPU** provides a sense of the amount of computer time for the computation of the smoother $\hat{\theta}(\epsilon_S)$. The computation is done on an IRIS Workstation and the random noise is generated by **DRNOR**$(\cdot)$, which is a subroutine in the SLATEC library and generates random numbers with the normal distribution of mean 0 and variance 1. The random design points $x_i \sim U(0, 1)$ are generated in the following way:

1. for $i = 1, \ldots, n$, generate $x_i :=$ **DRAND**(0) until $|x_i - x_j| \geq \frac{1}{2n}$ for all $1 \leq j < i$;
2. rearrange x_i's such that $x_i < x_{i+1}$ for $1 \leq i \leq n-1$.

Here **DRAND**$(\cdot)$ is the standard FORTRAN 77 function which generates uniformly distributed random numbers between 0 and 1. Note that we reject randomly generated x_i if it is too close to previously generated x_j's, since the smoothing result would not be very satisfactory if

$$\min_{2 \leq i \leq n} |x_i - x_{i-1}| / \max_{2 \leq i \leq n} |x_i - x_{i-1}|$$

were too small. We consider $\frac{1}{2n}$ as a reasonable lower bound for the minimum distance between two consecutive x_i's, since $\max_{2 \leq i \leq n} |x_i - x_{i-1}|$ could be $\frac{1}{2}$ in the worst case. Here we are not particularly interested in how the estimator will be affected by the design points. The purpose of including $x_i \sim U(0, 1)$ is to see whether unequally spaced design points will affect the smoothing process or not.

In summary, we can conclude from our numerical simulations that

1. $\mathbf{E_S} \cong \mathbf{E_I}$ and $\mathbf{R_S} \cong \mathbf{R_I}$ in all cases;
2. $\mathbf{R_I}$ is more or less in a neighborhood of 90%;
3. the smoother $\hat{\theta}(\epsilon_S)$ is not very sensitive to the variance of the noise or outliers;
4. the smoothing process is not affected by the location of x_i's as long as the ratio $\min_{2 \leq i \leq n} |x_i - x_{i-1}| / \max_{2 \leq i \leq n} |x_i - x_{i-1}|$ is not too small;
5. in all testing cases, $\frac{\mathbf{N}}{n} \leq 8$ and $\mathbf{N_0} \leq 5$; in general, $\frac{\mathbf{N}}{n}$ is very small and it only took a few seconds to find $\hat{\theta}(\epsilon_S)$;
6. the smoothing process can not effectively produce a satisfactory estimator with desirable shapes, if the underlying regression function $f(x)$ has a "fairly flat" segment.

The numerical results indicate that the computational cost of the regression scheme is about $\mathcal{O}(n^2)$ flops (cf. [24] and Table 1), which is far less expensive than we expected. More importantly, it seems very easy and effective to use $(m-1)$ (the number of sign changes of the second divided difference) to tune the smoothing process. First of all, one can always have a reasonable estimate of the number of inflection points that the original curve has. Even if the estimate does not produce a smoother with the desirable shape, it is easy to adjust m accordingly and recalculate the new smoother.

6. Conclusions and comments.

We have proposed a smoothing technique controlled by the number of convex/concave pieces of the fitting curve. The smoother is fairly easy to calculate, it has many good properties a smoother should have, and it has statistical properties similar to that of polynomial regression in some situations.

Extensive numerical simulations indicate that our piecewise convex/concave smoothing scheme does produce a fairly good piecewise convex/concave fitting in many different circumstances. However, the smoothing process might not be able to produce a satisfactory estimator with a desirable shape if the original regression function $f(x)$ has a "fairly flat" segment. A remedy to this deficiency might be an adaptive smoothing process as follows:

$$\begin{aligned} &\min \ \Sigma_{i=1}^{n}(y_i - \theta_i)^2 \\ &\text{subject to } l \leq \Delta_k\theta \leq u, \end{aligned}$$

where l and u are column vectors with $(n-k)$ components. Then, if necessary, one can make $l_i = u_i = 0$ for those x_i's in the x-range where $f(x)$ is supposed to be flat. From our simulation experiments, we notice that, in most cases, the smoother is not sensitive to an overestimation of m by 1. That is, if we increase m by 1 in our simulations, the mean squared errors will be more or less the same and, visually, one can hardly see the difference between the old and new smoothers.

It may be pointed out that the emphasis of this paper is on the computation of a piecewise convex/concave estimator. Much more is to be done about the smoothing procedure proposed here. For example, the monotonicity of $I(\hat{\theta}(\epsilon))$ as a function of ϵ, construction of a confidence band for the curve, perhaps using bootstrap techniques, and use of the method for prediction, etc., are yet to be explored. Note that Mammen [26] proved that, under some mild conditions on η_i which guarantee that the distribution of η_i does not have heavier tails than a Gaussian distribution,

$$\frac{1}{n}\sum_{i=1}^{n}(\hat{g}_n(x_i) - f(x_i))^2 \leq \mathcal{O}_p\left((mD)^{\frac{1}{2(k+1)}} n^{-\frac{k+1}{k+2}}\right),$$

where $\hat{g}_n$ is the solution to (3) and $A_n = \mathcal{O}_p(B_n)$ means that there exists a positive constant C such that, for any $\delta > 0$,

$$\Pr\left(\left|\frac{A_n}{B_n}\right| \leq C\right) \geq 1 - \delta$$

for sufficiently large n. It would be interesting to see whether similar results hold for (7) or (6).

The objective function considered in this article is the negative log likelihood function of normal errors. Hence the problem (9) can be viewed as a constrained maximum likelihood problem, that is, maximizing the likelihood function subject to certain constraints. One can take any one-dimensional exponential family of distributions instead of the normal distribution and maximize the likelihood function subject to the restrictions we have considered. Adoption of the algorithm to this case can be done, for example, by using sequential programming techniques (cf. [14]).

Acknowledgement. The first author would like to thank the Research Foundation of Old Dominion University for its support during the summer of 1992.

REFERENCES

[1] R. E. Barlow, D. J. Bartholomew, J. M. Bremner, and H. D. Brunk, *Statistical Inference Under Order Restriction—The Theory and Application of Isotone Regression*, John Wiley & Sons, NY, 1972.

[2] C. de Boor, *A Practical Guide to Splines*, Springer-Verlag, New York, 1978.

[3] W. S. Cleveland, *Robust locally weighted regression and smoothing scatter plots*, J. Amer. Statist. Assoc., 74 (1979), pp. 829-836.

[4] A. Cuevas and W. Gonzalez-Manteiga, *Data-driven smoothing based on convexity properties*, in Nonparametric Functional Estimation and Related Topics, G. Roussas ed., Kluwer Academic Publishers, the Netherlands, 1991, pp. 225-240.

[5] M. P. Cullinan, *Data smoothing using non-negative divided differences and l_2 approximation*, IMA J. Numer. Anal., 10 (1990), pp. 583-608.

[6] M. P. Cullinan and M. J. D. Powell, *Data smoothing by divided differences*, in Numerical Analysis Proc. Dundee 1981, G.A. Watson ed., LNIM 912, Springer-Verlag, Berlin, pp. 26-37.

[7] I. C. Demetriou, *Data Smoothing by Piecewise Monotonic Divided Differences*, Ph.D. thesis, University of Cambridge, Cambridge, UK, 1985.

[8] ———, *Best L_1 piecewise montone data modelling*, Internat. Trans. Opl. Res., 1 (1994), pp. 85-94.

[9] I. C. Demetriou and M. J. D. Powell, *Least squares smoothing of univariate data to achieve piecewise monotonicity*, IMA J. Numer. Anal., 11 (1991), pp. 411-432.

[10] ———, *The minimum sum of squares change to univariate data that gives convexity*, IMA J. Numer. Anal., 11 (1991), pp. 433-448.

[11] F. Deutsch, *The method of alternating orthogonal projections*, in Approximation Theory, Spline Functions and Applications, S. P. Singh, ed., Kluwer Academic Publishers, Boston, 1992, pp. 105-121.

[12] F. Deutsch and H. Hundal, *The Rate of Convergence of Dykstra's Cyclic Projections Algorithm: The Polyhedral Case*, preprint, 1993, Department of Mathematics, The Pennsylvania State University, University Park, PA.

[13] R. L. Dykstra, *An algorithm for restricted least squares regression*, J. Amer. Statist. Assoc., 78 (1983), pp. 837-842.

[14] A. V. Fiacco and G. P. McCormick, *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*, John Wiley, New York, 1968.

[15] J. Friedman, *A variable span smoother*, Department of Statistics Technical Report LCS5, Stanford University, Stanford, CA.

[16] I. J. Good and R. A. Gaskins, *Density estimation and bump-hunting by the penalized likelihood method exemplified by scattering and meteorite data*, J. Amer. Statist. Assoc., 75 (1980), pp. 43-73.

[17] D. L. Hanson and G. Pledger, *Consistency in concave regression*, Ann. Statist., 4 (1976), pp. 1038-1050.

[18] W. Härdle, *Asymptotic maximal deviation of M-smoothers*, J. Multi. Anal., 29 (1989), pp. 163-179.

[19] ———, *Applied Nonparametric Regression*, Econometric Society Monographs, No. 19, Cambridge University Press, Cambridge, UK, 1990.

[20] W. Härdle and A. Bowman, *Bootstraping in nonparametric regression: Local adaptive smoothing and confidence bands*, J. Amer. Statist. Assoc., 83 (1988), pp. 102-110.

[21] C. Hildreth, *Point estimates of ordinates of concave functions*, J. Amer. Statist. Assoc., 49 (1954), pp. 598-619.

[22] S. Holm and M. Frisén, *Nonparametric regression with simple curve characteristics*, Research Report 4, Department of Statistics, University of Güteborg, Germany, 1985.

[23] N. L. Johnson and S. Kotz, *Continuous Univariate Distributions*-1, John Wiley, New York, 1970.

[24] W. Li and J. Swetits, *A Newton method for convex regression, data smoothing, and quadratic programming with bounded constraints*, SIAM J. Optim., 3 (1993), pp. 466-488.

[25] Y. Y. Lin and J.-S. Pang, *Iterative methods for large quadratic programs: A survey*, SIAM J. Control Optim., 29 (1987), pp. 383-411.

[26] E. Mammen, *Nonparametric regression under qualitative smoothness assumptions*, Ann. Statist., 19 (1991), pp. 741-759.

[27] R. L. Matzkin, *Semiparametric estimation of monotone and concave utility functions for polychotomous choice models*, Econometrica, 59 (1991), pp. 1315-1327.

[28] D. R. Miller and A. Sofer, *Least-squares regression under convexity and higher-order difference constraints with application to software reliability*, in Advances in Order Restricted Inference, Lecture Notes in Statistics 33, Springer-Verlag, New York, 1986, pp. 91-124.

[29] F. Mosteller and J. W. Tukey, *Data Analysis and Regression*, Addison Wesley, NY, 1977.

[30] H.-G. Müller and J.-L. Wang, *Nonparametric analysis of changes in hazard rates for censored survival data: An alternative to change-point models*, Biometrika, 77 (1990), pp. 305-314.

[31] ———, *Locally adaptive hazard smoothing*, Probab. Theory Related Fields, 85 (1990), pp. 523-538.

[32] J. Orear and D. Cassel, *Applications of statistical inference to physics*, in Foundations of Statistical Inference, V. P. Godambe and D. A. Sprott, eds., Holt, Rinehart and Winston of Canada, Toronto, 1971, pp. 280-288.

[33] C. H. Reinsch, *Smoothing by spline functions*, Numer. Math., 10 (1967), pp. 177-183.

[34] J. A. Rice, *Bandwidth choice for nonparametric regression*, Ann. Statist., 12 (1984), pp. 1215-1230.

[35] T. Robertson, F. T. Wright, and R. L. Dykstra, *Order Restricted Statistical Inference*, Wiley Series in Probability and Mathematical Statistics, John Wiley, NY, 1988.

[36] I. J. Schoenberg, *Spline functions and the problem of graduation*, Proc. Nat. Acad. Sci., 52 (1964), pp. 947-950.

[37] B. W. Silverman, *Using kernel density estimates to investigate multimodality*, J. R. Statist. Soc., 43 (1981), pp. 97-99.

[38] ———, *Some properties of a test for multimodality based on kernel density estimates*, in Probability, Statistics and Analysis, J. F. C. Kingman and G. E. H. Reuter, eds., Cambridge University Press, Cambridge, UK, 1983, pp. 248-259.

[39] ———, *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, London, 1986.

[40] G. Wahba, *Spline Models for Observational Data*, Society for Industrial and Applied Mathematics, Philadelphia, PA 1991.

[41] G. Wahba and S. Wold, *A completely automatic French curve: Fitting spline functions by cross-validation*, Comm. Statist., Ser. A, 4 (1975), pp. 1-17.

[42] E. T. Whittaker, *On a new method of graduation*, Proc. Edinburgh Math. Soc., 41 (1923), pp. 63-75.

SIAM J. SCI. COMPUT.
Vol. 17, No. 2, pp. 538-559, March 1996

AN ANALYSIS OF APPROXIMATE NONLINEAR ELIMINATION*

PAUL J. LANZKRON†, DONALD J. ROSE‡, AND JAMES T. WILKES§

Abstract. We present a method for solving systems of nonlinear equations suitable for problems where convergence of an approximate Newton method is initially slow. The method, nonlinear elimination (NlEm), eliminates the nonlinear equations and appropriate variables deemed to be causing the problem. An analysis of the method is given and leads to a detailed algorithm that reduces automatically to an approximate Newton method near the root of the system of nonlinear equations. Several examples are given that demonstrate the efficacy of the method.

Key words. nonlinear equations, global methods, approximate Newton method

AMS subject classifications. 65H10, 65H20

1. Introduction.

We consider the numerical solution of a system of nonlinear equations

$$g(w) = 0, \tag{1.1}$$

where $g = (g_1, g_2, \ldots, g_n)^T$ and $w = (w_1, w_2, \ldots, w_n)^T$. Generically we consider the methodology of [1] and the analytic framework presented there. This setting ensures that for any w_0 in some set S, a sequence of iterates w_k will converge to $w^* \in S$ and $g(w^*) = 0$. The iterates are defined as

$$w_{k+1} = w_k + t_k x_k, \tag{1.2}$$

where x_k *approximates* z_k in

$$g'_k z_k = -g_k, \tag{1.3}$$

$g'_k \equiv g'(w_k)$, and $g_k \equiv g(w_k)$, and $t_k \in (0, 1]$ is chosen to force

$$\|g_{k+1}\| < \theta \|g_k\| \tag{1.4}$$

for some $0 < \theta < 1$. The sense in which x_k approximates z_k of (1.3) is measured by

$$\alpha_k = \|g'_k x_k + g_k\| / \|g_k\| \tag{1.5}$$

and all $\alpha_k \leq \alpha_0 < 1$ suffices for convergence (for some sequence of t_k). We call this algorithmic approach GAN, for global approximate Newton, "global" referring to the set S which can be $\mathbb{R}^n$ under appropriate conditions.

Such variations of Newton's method have been used successfully in significant technology areas including circuit and device simulation [2, 5]. The trick in any of these applications is in picking an "inner" solver for (1.3) to make $\alpha_k \to 0$ and $t_k \to 1$ to ensure (1.4) and superlinear convergence. This can be particularly challenging in practice since we often cannot be sure a priori that the sufficient convergence conditions are satisfied, nor can we wait for $k \to \infty$.

The use of nonlinear elimination (NlEm, en-lem) is motivated by problems in which convergence seems to be interminably tedious and yet there is no evidence of ill conditioning (or singularity). This leads us to believe that the nonlinearities in g are unbalanced, by which

*Received by the editors July 6, 1995; accepted for publication (in revised form) October 22, 1994.

†Raytheon Company, 1001 Boston Post Road, Marlborough, MA 01752 (pjl@tif396.ed.ray.com). The work of this author was supported in part by the National Science Foundation under grant NSF-CCR-92-09444.

‡Department of Computer Science, Duke University, Durham, NC 27708 (djr@cs.duke.edu). The work of this author was supported in part by the Office of Naval Research under grant N00014-89-J-1644.

§Department of Mathematical Sciences, Appalachian State University, Boone, NC 28608 (jtw@cs.appstate.edu).

we mean some "subfunction" $g_1(u, v)$ regarded as a function of u given v causes t_k to be small. We propose to eliminate g_1 as an inner iteration.

Consider $g(w)$ to be partitioned as

$$\begin{aligned} &g(w) = (g_1(u, v), g_2(u, v))^T, \\ &w = (u, v)^T. \end{aligned} \tag{1.6}$$

This leads to a block Jacobian

$$g'(u, v) = \begin{bmatrix} g_{11}(u, v) & g_{12}(u, v) \\ g_{21}(u, v) & g_{22}(u, v) \end{bmatrix}, \tag{1.7}$$

where $g_{ij}(u, v) = \frac{\partial g_i}{\partial g_j}(u, v)$. For conciseness we write $g_{ij} \equiv g_{ij}(u, v)$.

For smooth $g_1(u, v)$ the solvability of $g_1(u, v) = 0$ for v in an appropriate set leads to an implicit function $h(v)$ such that

$$g_1(h(v), v) = 0. \tag{1.8}$$

Differentiating (1.8) as a function of v leads to

$$g_{11}h'(v) + g_{12} = 0, \tag{1.9}$$

and if $g_{11}(h(v), v)$ is nonsingular, then

$$h'(v) = -g_{11}^{-1}g_{12}. \tag{1.10}$$

Indeed the operational equation (1.10) and the existence of the unique mapping $h(v)$ constitute the essence of the implicit function theorem, see [12, Thm. 5.2.4], which requires the nonsingularity of g_{11}.

Assuming that $u = h(v)$ exists on an appropriate set, we attempt to solve equation (1.1), which can be rewritten as

$$f(v) = 0, \tag{1.11}$$

where $f(v) \equiv g_2(h(v), v)$ for v using GAN. To use GAN we must compute the Jacobian $f'(v)$; differentiating (1.11) yields

$$f'(v) = g_{21}h'(v) + g_{22}, \tag{1.12}$$

where g_{21}, g_{22} are evaluated at $(h(v), v)$. Making the substitution (1.10), we obtain

$$f'(v) = g_{22} - g_{21}g_{11}^{-1}g_{12} \tag{1.13}$$

and we recognize the right-hand side of (1.13) as a Schur complement. More precisely we see that the Newton direction equation associated with (1.11), namely,

$$f'(v)\Delta v = -f(v), \tag{1.14}$$

can be embedded into the larger matrix equation

$$\begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta v \end{bmatrix} = \begin{bmatrix} -g_1 \\ -g_2 \end{bmatrix}, \tag{1.15}$$

where all functions are evaluated at $(h(v), v)$ and $g_1(h(v), v) = 0$. Equation (1.14) arises from block Gaussian elimination on (1.15) as indicated by (1.13). This is computationally attractive since nonlinear elimination leads to the same algebra as GAN on the whole system, but now applied at the point $(h(v), v)$. To summarize there are two nonlinear solve processes: an "inner" equation solve to evaluate $h(v)$ from (1.8) and an "outer" Newton iteration to solve (1.11) via (1.15). When $h(v)$ is evaluated "exactly," i.e., $g_1(h(v), v) = 0$, then we show (Theorem 4.2) that the outer GAN (1.15) converges quadratically to the solution $g_2(h(v), v) = 0$ as expected.

Note that when a sparse matrix package can be used for GAN, it is also applicable for NlEm. That is, starting at the point $(h(v), v)$, both NlEm and GAN would need to solve the system (1.15). It is unimportant that the variables be ordered so that the Δu variables are first. In fact it is easy to imagine cases where such an ordering would lead to significant fill. After the linear system is solved the Δv variables can be gathered easily.

Many investigators have studied nonlinear elimination for particular applications. We discuss two of these applications here. The first is macromodeling circuits. A circuit can be thought of as a k-terminal device. In modeling the circuit the user wishes to know how changing the voltage at some of the input terminals affects the voltage at other terminals. The number of unknowns for the system is then k plus the number of internal unknowns in the device. In [6] the authors show how to eliminate the internal unknowns from the system of nonlinear equations that must be solved at each time step. More recent work on applying NlEm to circuit simulation is reported in [13].

A second widely used, although only tangentially connected, application of nonlinear elimination is in nonlinear least squares. Consider the problem of finding (u, v) such that

$$\|y - g(u, v)\|_2 \tag{1.16}$$

is minimized, where $y \in \mathbb{R}^{n\times 1}$ is a set of observations and $g : \mathbb{R}^p \to \mathbb{R}^n$ is nonlinear. Consider the case where

$$g(u, v) \equiv l(v)u = \begin{bmatrix} g_{11}(v) \\ g_{21}(v) \end{bmatrix} u. \tag{1.17}$$

When v is fixed in equation (1.17), equation (1.16) turns into a linear least squares problem. The problem given in (1.16) may then be rewritten as

$$\min_v \|y - g(h(v), v)\|_2, \tag{1.18}$$

where

$$h(v) = \left(l(v)^T l(v)\right)^{-1} l(v)^T y. \tag{1.19}$$

We have assumed, as in [10], that $l(v)$ has full column rank. The case when $l(v)$ does not have full column rank is handled in [7, 8]. The difference between this approach and our approach is that we eliminate variables *and* equations so that $f(v)$ is still a system of p nonlinear equations in p unknowns.

Nonlinear elimination arises naturally in constrained optimization and can be viewed in the context of Lagrange multipliers. Most authors do not choose such an exposition; see [3, p. 141] for a related discussion. We remark also that Brown's method [4] can be regarded as a specialization of nonlinear elimination as discussed in [12, NR 7.14-15, pp. 227-229].

This paper is organized as follows. In §2 the NlEm algorithm is presented along with a simple example. Sections 3-5 contain an analysis of the whole procedure, presented as constructively as possible. Since we suggest that NlEm can be part of a general algorithmic strategy for solving nonlinear systems, this analysis presents conditions that ensure convergence in a way that is compatible with the theory presented in [1]. Section 4 discusses the

```
Input: $w_0 = (u_0, v_0)$ and $g$.
1.  Choose $g_1$ equations splitting $g(w)$ into $g_1(u, v)$ and $g_2(u, v)$.
    By $g_1(u, v)$ we mean that if $g_1 : \mathbb{R}^n \rightarrow \mathbb{R}^j$, then $u \in \mathbb{R}^j$.
2.  Given the initial $v_0$, solve $g_1(h(v_0), v_0) = 0$ for $h(v_0)$.
3.  $k \leftarrow 0$
4.  repeat until convergence
5.      Solve
        $\begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix} \begin{bmatrix} \Delta u_k \\ \Delta v_k \end{bmatrix} = \begin{bmatrix} -g_1 \\ -g_2 \end{bmatrix}$,
        with $g_{11}, g_{12}, g_{21}, g_{22}$, and $g_2$ evaluated at $(h(v_k), v_k)$.

6.      Find $t_k \in (0, 1]$ such that
        $\|g(h(v_k + t_k\Delta v_k), v_k + t_k\Delta v_k)\| < \theta\|g(h(v_k), v_k)\|$
        Note: a solve for $u = h(v_k + t_k\Delta v_k)$ of
        $g_1(u, v_k + t_k\Delta v_k) = 0$
        is required for every $t_k$.

7.      $v_{k+1} = v_k + t_k\Delta v_k$
        Note: $h(v_{k+1})$ is implicitly determined in step 6.
```

FIG. 1. *Outline of the NlEm algorithm.*

case when (1.8) is solved exactly and §5 discusses how accurately (1.8) must be solved to retain the higher-order convergence of Newton's method. The analysis in §5 leads to a refined algorithm in §6 that unifies NlEm and GAN into a single methodology. Section 7 contains several computed examples.

2. Implementation overview. In this section we give a brief overview of how to implement the NlEm algorithm along with an illustrative example. In §6, a detailed algorithm is presented that unifies GAN and NlEm.

2.1. NlEm algorithm. The outline of the algorithm is given in Figure 1.

Often the hardest part of the implementation is in step 1, since determining the set of equations that belong in g_1 is usually problem dependent. Once the set is chosen the solve in step 2 moves the initial point (u_0, v_0) to a new point in the domain $(h(v_0), v_0)$. There may be an increase in the overall norm of g at this new point; i.e.,

$$\|g(h(v_0), v_0)\| > \|g(u_0, v_0)\|. \tag{2.1}$$

However, the norm of g decreases monotonically thereafter under appropriate conditions. There are times when the set of g_1 equations may be changed dynamically after the algorithm has been started. However, arbitrarily changing the set of equations represented by g_1 could lead to thrashing (i.e., the norm of g goes down for a while, but the g_1 equations are changed causing $\|g\|$ to be as large or larger than previously).

The computation in step 6 implicitly requires a solve of the g_1 equation. That is, to compute $h(v_k + t_k\Delta v_k)$ the g_1 equation must be solved. Step 6 is potentially expensive, since every new choice of t_k requires a solve of the g_1 equations. Note that in step 6 we are trying to make the overall norm of g go down. This is necessary since $g_1 \neq 0$ in practice. When $g_1 \neq 0$ we only have an approximation to $h(v)$, say $\tilde{h}(v)$. Thus, $g_2(h(v), v)$ is approximated

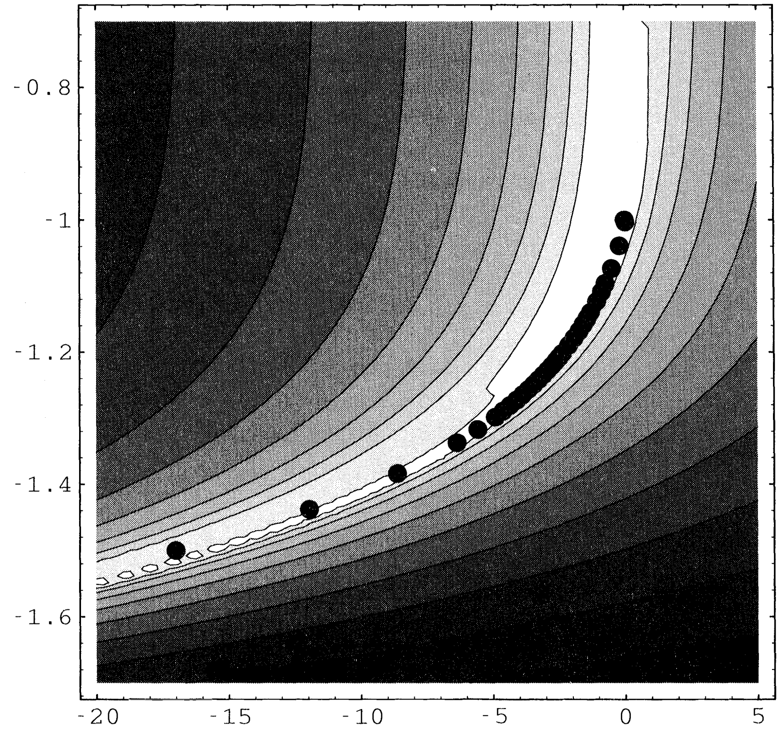


FIG. 2. *Convergence of GAN overlaid on* $\log(\|f\| + 1)$.

by $g_2(\tilde{h}(v), v)$. A decrease in $\|g_2(\tilde{h}(v), v)\|$ does not imply that $\|g_2(h(v), v)\|$ has decreased; see §5. Therefore, we enforce norm reduction on all of g.

2.2. A simple illustrative example. We believe NlEm will be helpful when the system of equations has some badly scaled components. The poor scaling is not in terms of normal linear misscaling, rather it is due to the nonlinearity of the problem. In this section, we present a simple example demonstrating how NlEm helps.

Consider the system f of two equations given by

$$\begin{aligned} f_1(u, v) &= (u - v^7)^3 + v^3, \\ f_2(u, v) &= (u + v)^{\frac{1}{5}} + 1. \end{aligned} \tag{2.2}$$

When viewed in terms of norms the curve $u = v^7 - v$ is close to the bottom of a very steep valley. A GAN iterate in the valley causes the Newton direction to point tangent to the valley. Since the valley curves and the Newton direction points in a direction nearly tangent to the bottom of the valley, only small values of the damping parameter can be chosen; larger values give an increase in $\|g\|$. In Figure 2 the starting point $(-17.085938, -1.5)$ leads the GAN iterates into the steep curved valley. GAN takes 32 steps to converge to the solution $(0, -1)$.

To use NlEm we must first decide on the equation to eliminate. For this problem it is clear that the first equation is causing the difficulties; that is, movement away from the curve $u = v^7$ causes a large increase in the norm of the system. This is not true for the second equation. Having decided on the equation to eliminate, we must then decide on the variable to be eliminated with it; in this case we choose to eliminate the u variable. In realistic problems the variable eliminated will be more obvious. For a given v we will solve for a u such that

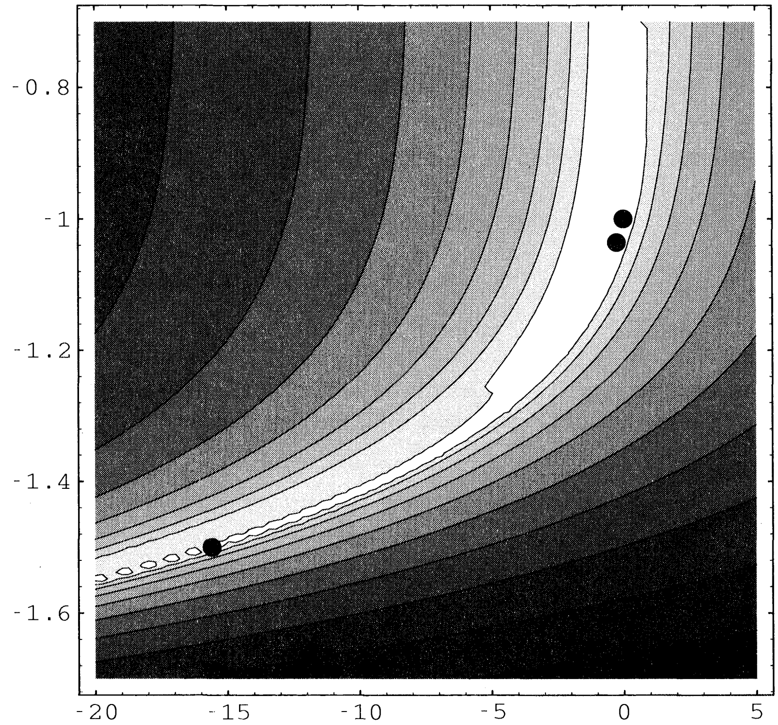


FIG. 3. *Convergence of NlEm overlaid on* $\log(\|f\|+1)$.

$g_1(u, v) = 0$. Solving for u in the first equation means that $u = h(v) = v^7 - v$. The eliminated system, $g_2(h(v), v) = v^{7/5} + 1$, is a much simpler problem to solve. In Figure 3 we see that the number of NlEm iterations from the same starting point drops to three.

3. Preliminaries. Recall that our objective is to solve (1.11) using Newton's method. This section gives a review of the GAN convergence theory and provides results used in later sections.

3.1. Analysis of GAN. The following analysis of GAN is derived from [1]. Let $g_k \equiv g(w_k)$ with w_0 some initial guess for the solution. There are three basic assumptions used to prove convergence of GAN.

Assumption A1. The closed level set

$$S_0 = \{w \mid \|g(w)\| \leq \|g_0\|\} \tag{3.1}$$

is bounded.

The following assumption is equivalent to Assumption A2 presented in [1].

Assumption A2. g is differentiable, the Jacobian $g'(w)$ is continuous and nonsingular on S_0, and the sequence $\|x_k\|$ is uniformly bounded, i.e.,

$$\|x_k\| \leq k_1\|g_k\| \tag{3.2}$$

for $k_1 \geq 0$ and $w_k \in S_0$.

Assumption A3. The Jacobian g' is Lipschitz; i.e.,

$$\|g'(y) - g'(z)\| \leq k_2\|y - z\| \tag{3.3}$$

for

$$y, z \in S_1 = \{y \mid \|y\| \leq \sup_{z \in S_0} \|z\| + k_1\|g_0\|\}. \tag{3.4}$$

To continue the analysis of GAN we define the quantities

$$\begin{aligned} A_k &= (g_k + g_k' x_k)/\|g_k\|, \\ B_k &= \left\{ \int_0^1 \left[g'(w_k + st_k x_k) - g_k' \right] t_k x_k ds \right\} / \, (t_k\|g_k\|)^2, \\ \alpha_k &= \|A_k\|, \qquad \beta_k = \|B_k\|, \end{aligned} \tag{3.5}$$

where we have assumed $g_k \neq 0$. The quantity α_k defines the relative size of the linear residual.

Following the analysis of [1], the mean value theorem yields

$$g_{k+1} = g_k + \int_0^1 g'(w_k + st_k x_k)(w_{k+1} - w_k) ds. \tag{3.6}$$

Adding and subtracting $g_k'(w_{k+1} - w_k)$ on the right-hand side yields

$$g_{k+1} = g_k + g_k'(w_{k+1} - w_k) + \int_0^1 \left[g'(w_k + st_k x_k) - g_k' \right] (w_{k+1} - w_k) ds. \tag{3.7}$$

Recalling that $w_{k+1} = w_k + t_k x_k$ we have

$$g_{k+1} = (1 - t_k) g_k + t_k A_k \|g_k\| + t_k^2 B_k \|g_k\|^2. \tag{3.8}$$

And finally taking norms we are left with

$$\|g_{k+1}\| \leq \|g_k\| \left((1 - t_k) + t_k \alpha_k + t_k^2 \beta_k \|g_k\| \right). \tag{3.9}$$

Under Assumptions A1-A3, β_k can be shown to be bounded, $\beta_k \leq k_1^2 k_2/2$. It is usually possible to control the size of α_k by computing x_k more accurately. This is the case when an iterative method is being used to solve the Newton equations. It is clear from (3.9) that if $\alpha_k < 1$ then there exists a t_k such that $\|g_{k+1}\| < \|g_k\|$.

Convergence of GAN is given by the following proposition and theorem.

PROPOSITION 3.1 (see [1]). *Assume that* A1-A3 *hold and that all* $\alpha_k \leq \alpha_0$. *If* t_k *is chosen appropriately, then*

1. *all* $w_k \in S_0$, *the sequence* $\|g_k\|$ *is strictly decreasing, and* $\|g_k\| \to 0$; *furthermore,*
2. $\|g_{k+1}\|/\|g_k\| \to 0$ *iff* $\alpha_k \to 0$ *and for any fixed* $p \in (0, 1]$,

$$\|g_{k+1}\| \leq c_1 \|g_k\|^{1+p}, \tag{3.10}$$

iff

$$\alpha_k \leq c_2 \|g_k\|^p \tag{3.11}$$

for positive constants c_1 *and* c_2.

THEOREM 3.2 (see [1]). *Under the conditions of Proposition* 3.1,

1. *there exists a* $w^* \in S_0$ *with* $w^* = \lim w_k$ *and* $g(w^*) = 0$;
2. *on* S_0 *the convergence of* $\{w_k\}$ *to* w^* *is superlinear or Q-order* $(p + 1)$ *if* $\alpha_k \to 0$ *or* $\alpha_k \leq c_2 \|g_k\|^p$, *respectively.*

3.2. Additional results. We conclude this section with results that will be used in subsequent sections.

PROPOSITION 3.3. *Assume that GAN starting from w_0 is convergent to w^* with $g(w^*) = 0$. Let $\|g'^{-1}(w)\| \leq M$, for $u \in S_0$, and $\|g_{i+1}\| \leq \theta \|g_i\|$ for every i, where $g_i \equiv g(w_i)$ and $\theta < 1$. Then*

$$\|w_0 - w^*\| \leq \frac{1}{1-\theta} M \|g(w_0)\|. \tag{3.12}$$

Proof. Since GAN is assumed to be convergent, we know that $w_i \in S_0$ for all i. Thus.

$$\begin{aligned}\|w_0 - w^*\| &= \left\| w_0 - \left(w_0 + \sum_0^\infty t_i \Delta w_i \right) \right\| = \left\| \sum_0^\infty t_i \Delta w_i \right\| \\ &\leq \sum_0^\infty \|\Delta w_i\| = \sum_0^\infty \| - g_i'^{-1} g_i \| \\ &\leq M \sum_0^\infty \|g_i\| \leq M \|g_0\| \sum_0^\infty \theta^i = \frac{1}{1-\theta} M \|g(w_0)\|. \qquad \square\end{aligned}$$

The proofs in this paper are complicated somewhat by the changes in sizes of norms. We will address this problem by restricting the matrix norms we consider to p-norms. These norms have the following properties.

Property P1. Let $u \in \mathbb{R}^m$, and $v = [0, 0, \ldots, 0, u]^T \in \mathbb{R}^n$; then

$$\|u\|_p = \|v\|_p. \tag{3.13}$$

For example,

$$\|u\|_p = \left(\sum_{i=1}^m w_i^p \right)^{\frac{1}{p}} = \left(\sum_{i=1}^m w_i^p + \sum_{i=m+1}^n 0^p \right)^{\frac{1}{p}} = \|v\|_p. \tag{3.14}$$

Note that (3.13) holds however the elements of u are distributed in v.

Property P2. Let $u \in \mathbb{R}^m$, $v = [u, 0, 0, \ldots, 0]^T \in \mathbb{R}^n$ and $w = [u, y]^T \in \mathbb{R}^n$; then

$$\|v\|_p \leq \|w\|_p. \tag{3.15}$$

The following lemma can easily be verified to show that Properties P1 and P2 carry through to matrices.

LEMMA 3.4. *Let $A \in \mathbb{R}^{j \times k}$, $j, k \leq n$; then*

$$\|A\|_p = \left\| \begin{bmatrix} 0 & 0 \\ 0 & A \end{bmatrix} \right\|_p \leq \left\| \begin{bmatrix} B & C \\ D & A \end{bmatrix} \right\|,$$

where $\left[\begin{smallmatrix} B & C \\ D & A \end{smallmatrix}\right] \in \mathbb{R}^{n \times n}$.

We will refer to (3.13) as Property P1, and (3.15) as Property P2 without distinguishing between matrices and vectors.

LEMMA 3.5. *Let the nonsingular $A \in \mathbb{R}^{n \times n}$ be partitioned as*

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

with $A_{11} \in \mathbb{R}^{j\times j}$ *and nonsingular. If* $\|A^{-1}\| < k$ *and properties* P1 *and* P2 *hold for* $\|\cdot\|$, *then*

$$\|(A_{22} - A_{21}A_{11}^{-1}A_{12})^{-1}\| < k. \tag{3.16}$$

Proof.

$$\begin{aligned} \|(A_{22} - A_{21}A_{11}^{-1}A_{12})^{-1}\| &= \left\| \begin{bmatrix} 0 & 0 \\ 0 & (A_{22} - A_{21}A_{11}^{-1}A_{12})^{-1} \end{bmatrix} \right\| \\ &\le \|A^{-1}\| \\ &< k. \quad \square \end{aligned} \tag{3.17}$$

PROPOSITION 3.6. *If* g *is Lipschitz on a bounded set* S, *then* $\|g\|$ *is bounded.*

Proof. Since S is bounded, there exists $k \in \mathbb{R}^+$ such that $\|w - y\| \le k$ for all $w, y \in S$. Let w_0 be a particular member of S with $\|g(w_0)\| = C$. Then, for any $w \in S$, we have

$$\|g(w)\| \le \|g(w) - g(w_0)\| + \|g(w_0)\| = L\|w - w_0\| + C = Lk + C, \tag{3.18}$$

where L is the Lipschitz constant. □

PROPOSITION 3.7. *Let* $M(w) \in \mathbb{R}^{n\times n}$ *with* $M(w)$ *Lipschitz on some convex set with Lipschitz constant* k_0. *If* $M(w)$ *is partitioned as*

$$\begin{bmatrix} M_{11}(w) & M_{12}(w) \\ M_{21}(w) & M_{22}(w) \end{bmatrix} \tag{3.19}$$

with $M_{11}(w) \in \mathbb{R}^{j\times j}$ *and* $\|M_{11}^{-1}(w)\| \le k_1$, *then the Schur complement of* $M(w)$ *is also Lipschitz with Lipschitz constant*

$$k_0(1 + k_1k_2)^2, \tag{3.20}$$

where $\|M(w)\| \le k_2$.

Proof. Applying Proposition 3.6 gives us $\|M(w)\| \le k_2$. Using Properties P1 and P2, we know that $\|M_{ij}\| \le k_2$ for $i, j = 1, 2$. Let $A \equiv M(w_1)$ and $B \equiv M(w_2)$.

$$\begin{aligned} &\|\left(A_{22} - A_{21}A_{11}^{-1}A_{12}\right) - \left(B_{22} - B_{21}B_{11}^{-1}B_{12}\right)\| \\ &\le \|A_{22} - B_{22}\| \\ &\quad + \|B_{21}B_{11}^{-1}B_{12} - B_{21}B_{11}^{-1}A_{12} + B_{21}B_{11}^{-1}A_{12} - A_{21}A_{11}^{-1}A_{12}\| \\ &\le k_0\|x - y\| + \|B_{21}B_{11}^{-1}\left(B_{12} - A_{12}\right)\| + \|\left(B_{21}B_{11}^{-1} - A_{21}A_{11}^{-1}\right)A_{12}\| \\ &\le k_0\|x - y\| + k_2k_1k_0\|x - y\| \\ &\quad + \|\left(B_{21}B_{11}^{-1} - A_{21}B_{11}^{-1} + A_{21}B_{11}^{-1} - A_{21}A_{11}^{-1}\right)A_{12}\| \\ &\le k_0\|x - y\| + k_2k_1k_0\|x - y\| \\ &\quad + k_2\|\left(B_{21} - A_{21}\right)B_{11}^{-1}\| + k_2\|A_{21}\left(B_{11}^{-1} - A_{11}^{-1}\right)\| \\ &\le k_0\|x - y\| + k_2k_1k_0\|x - y\| + k_2k_1k_0\|x - y\| \\ &\quad + k_2^2\|B_{11}^{-1}\|\|A_{11}^{-1}\|\|\left(B_{11} - A_{11}\right)\| \\ &\le k_0\|x - y\| + 2k_2k_1k_0\|x - y\| + k_2^2k_1^2k_0\|x - y\| \\ &= k_0(1 + k_1k_2)^2\|x - y\|. \quad \square \end{aligned} \tag{3.21}$$

4. Exact NlEm. The exact NlEm algorithm is Newton's method applied to the equation

$$f(v) \equiv g_2(h(v), v) = 0. \tag{4.1}$$

To show that the algorithm converges, we need to show that, under appropriate conditions, $h(v)$ exists and that the basic Assumptions A1–A3 hold on the function $f(v)$. These assumptions and making $\alpha_k \to 0$ ensure that Proposition 3.1 and Theorem 3.2 hold.

First, assume that GAN is convergent on g and that from any point (u, v) in a given set, GAN converges on g_1 giving $g_1(\hat{u}, v) = 0$ where $\hat{u} = h(v)$. For clarity sake, let $S_0(g)$ be the set S_0 given in (3.1) and let $S_1(g)$ be the set S_1 given in (3.4). Because of the partitioning of g, projections of sets are needed. In particular, let

$$S_1^2(g) = \{v \mid (u, v) \in S_1(g) \quad v \in \mathbb{R}^{n-j}\}. \tag{4.2}$$

The following assumptions are required for proof of the convergence of GAN applied to (4.1).

Assumption A1 remains unchanged and for Assumption A2 we make the stronger restriction that $\|g'^{-1}\| \le k_1$ over $S_0(g)$. Assumption A3 is modified below.

Assumption A4. Let

$$\|\hat{g}_1\| = \max_{w \in S_1(g)} \|g_1(w)\|; \tag{4.3}$$

then the set

$$S_0(g_1) = \{(u, v) \mid v \in S_1^2(g) \text{ and } \|g_1(u, v)\| \le \|\hat{g}_1\|\} \tag{4.4}$$

is bounded.

Assumption A5. The function g_1 is differentiable and g_{11} is continuous and nonsingular on $S_0(g_1)$, and

$$\|g_{11}^{-1}(w)\| \le k_5, \qquad w \in S_0(g_1). \tag{4.5}$$

Since convergence of the g_1 system is required only on the projection of $S_0(g_1)$, i.e., $g_1(u, v) = 0$ is solved with v fixed; the projection sets of $S_0(g_1)$ are defined as

$$S_0^1(g_1) = \{u \mid (u, v) \in S_0(g_1), \qquad u \in \mathbb{R}^j\}, \tag{4.6}$$

$$S_0^2(g_1) = \{v \mid (u, v) \in S_0(g_1), \qquad v \in \mathbb{R}^{n-j}\}. \tag{4.7}$$

This leads to a modification of Assumption A3 given previously.

Assumption A3. The Jacobian g' is Lipschitz with Lipschitz constant k_2 on

$$S_1(g_1) = \{(u, v) \mid v \in S_0^2(g_1) \text{ and } \|u\| \le \max_{w \in S_0^1(g_1)} \|w\| + k_5\|\hat{g}_1\|\}, \tag{4.8}$$

where $\|\hat{g}_1\|$ is defined as before.

The sets $S_0(g)$, $S_1(g)$, $S_0(g_1)$, and $S_1(g_1)$ are illustrated in two dimensions in Figure 4; the u-axis is horizontal and the v-axis is vertical. Note that $S_0(g)$ need not be connected. Under Assumptions A1–A5, there is a root of g in both of the disconnected regions. The sets $S_1(g)$ and $S_1(g_1)$ are both convex, but the set $S_1(g_1)$ has a capsule shape because it is stretched in the direction of the eliminated variable.

These sets come into play during various phases of the solution algorithm. Initially w_k is in $S_0(g)$. The approximate Newton direction is computed and a potential w_{k+1} produced. The potential $w_{k+1} = w_k + t_k x_k$ may be outside $S_0(g)$, but is within $S_1(g)$ (of course, when the final w_{k+1} is determined it must be in $S_0(g)$). At this point in the algorithm, a root u is sought for $g_1(u, v)$ from a starting point in $S_1(g)$. This root will be somewhere in $S_0(g_1)$. The Newton method used to find that root may generate points in $S_1(g_1)$ (in the same way as $w_k + t_k x_k$ may be outside of $S_0(g)$).

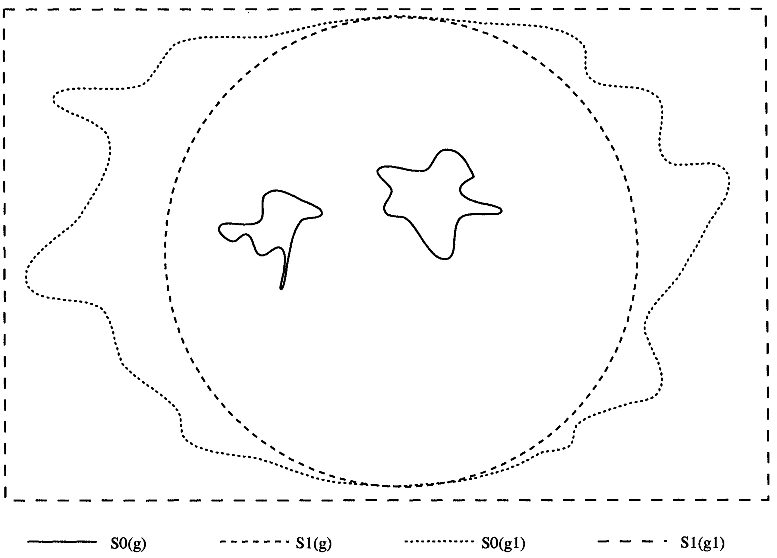


FIG. 4. *Illustration of the sets $S_0(g)$, $S_1(g)$, $S_0(g_1)$, and $S_1(g_1)$ in two dimensions.*

LEMMA 4.1. *Consider Assumptions* A3–A5. *For $v \in S_1^2(g)$, there is a $u \in S_0^1(g_1)$ such that*

$$g_1(u, v) = 0. \tag{4.9}$$

Thus, $h(v)$ exists for $v \in S_1^2(g)$ and

$$h'(v) = -g_{11}^{-1}(h(v), v)g_{12}(h(v), v). \tag{4.10}$$

Proof. Let

$$\|g'(w_1) - g'(w_2)\| \le k_2\|w_1 - w_2\| \tag{4.11}$$

hold for $\|\cdot\|$ satisfying Properties P1 and P2. Let $w_1 = (u_1, v)$ and $w_2 = (u_2, v)$. It follows that

$$\|g_{11}(w_1) - g_{11}(w_2)\| \le k_2\|u_1 - u_2\|. \tag{4.12}$$

Assumptions A4 and A5 and (4.12) represent the conditions for Proposition 3.1 and Theorem 3.2 to hold. Thus $g_1(u, v)$ has a root for fixed $v \in S_0^1(g_1)$. The remainder of the lemma follows from the implicit function theorem. □

THEOREM 4.2. *Consider Assumptions* A1–A5. *If $(u_0, v_0) \in S_0(g)$ with $g_1(u_0, v_0) = 0$, and at every iteration* (1.15) *is solved exactly, then GAN converges to the solution of $f(v) = 0$ and, furthermore, convergence is quadratic* (*Q-order* 2).

Proof. The proof will show that Assumptions A1–A3 hold for $f(v)$ and that $\alpha_k = 0$ (α_k as defined in (3.5)).

(A1) Let $z \in S_0(f) = \{v \| \|f(v)\| \le \|f(v_0)\|\}$, then by Property P1,

$$\|g(h(z), z)\| = \|f(z)\| \le \|f(v_0)\| = \|g(h(v_0), v_0)\|. \tag{4.13}$$

Thus, $(h(z), z) \in S_0(g)$ and by Property P2, $S_0(f)$ is bounded.

(A2) Since $(h(z), z) \in S_0(g)$ for $z \in S_0(f)$, we know that

$$\|\Delta v_k\| \leq \left\|\begin{bmatrix} \Delta u_k \\ \Delta v_k \end{bmatrix}\right\| = \|x_k\| \leq k_1\|g(h(v_k), v_k)\| = k_1\|f(v_k)\|, \tag{4.14}$$

using Properties P1 and P2 where appropriate.

Define

$$S_1(f) \equiv \{v \mid \|v\| \leq \sup_{w \in S_0(f)} \|w\| + k_1\|f(v_0)\|\}. \tag{4.15}$$

(A3) From Assumption A5 $g_{11}^{-1}(h(v), v)$ exists and is bounded. Note that $S_1(f) \subset S_0^2(g_1)$ and $g_1(h(v), v) = 0$ for $v \in S_0^2(g_1)$; thus

$$(h(v), v) \in S_0(g_1) \subset S_1(g_1). \tag{4.16}$$

These observations and Proposition 3.7 gives

$$\|f'(v_1) - f'(v_2)\| \leq k_2(1 + k_5k_3)^2\|(h(v_1), v_1)^T - (h(v_2), v_2)^T\| \tag{4.17}$$

for $v_1, v_2 \in S_1(f)$.

By the triangle inequality and Property P1 it follows that

$$\left\|\begin{bmatrix} h(v_1) - h(v_2) \\ v_1 - v_2 \end{bmatrix}\right\| \leq \|v_1 - v_2\| + \|h(v_1) - h(v_2)\|. \tag{4.18}$$

Since bounded derivative implies Lipschitz continuous (see Theorem 3.2.4 [12]) and

$$\|h'(v)\| = \| - g_{11}^{-1}(h(v), v)g_{12}(h(v), v)\| \leq k_5k_3, \tag{4.19}$$

we know that

$$\|h(v_1) - h(v_2)\| \leq k_7\|v_1 - v_2\|. \tag{4.20}$$

for some k_7.

Putting these results together gives

$$\|f'(v_1) - f'(v_2)\| \leq \left(k_2(1 + k_5k_3)^2(1 + k_7)\right) \|v_1 - v_2\| = k_8\|v_1 - v_2\|. \tag{4.21}$$

Finally, since in the exact case $\Delta v_k = -f_k'^{-1} f_k$, it follows that $\alpha_k = 0$. Therefore, Proposition 3.1 and Theorem 3.2 hold for $f(v)$. □

5. Approximate NlEm. In an approximate Newton method, the Newton correction is computed by

$$M_k x_k = -g_k, \tag{5.1}$$

where M_k is an approximation of g_k'. Convergence of this method is given by showing that $\alpha_k < 1$ in (3.5).

Recall that NlEm can be viewed as GAN applied to (1.11). The α_k for this nonlinear system is

$$\alpha_k = \frac{\|f(v_k) + f'(v_k)\Delta v_k\|}{\|f(v_k)\|}. \tag{5.2}$$

In approximate NlEm, (1.15) and $g_1(u, v_k) = 0$ are not solved exactly. Since the g_1 system is not solved exactly, the actual $h(v_k)$ is not computed at iteration k. Thus, verifying $\alpha_k < 1$ is impossible because the actual f and f' are not computed. Denote by $\tilde{h}(v)$ the computed approximation to $h(v)$. Although $h(v)$ was continuous and differentiable, $\tilde{h}(v)$ may be neither. In the remainder of this section we will use $\tilde{g}_k, \tilde{g}'_k, \tilde{g}_{1k}, \tilde{g}_{2k}, \tilde{g}_{11k}, \tilde{g}_{12k}, \tilde{g}_{21k}, \tilde{g}_{22k}$ to denote $g, g', g_1, g_2, g_{11}, g_{12}, g_{21}, g_{22}$, respectively, evaluated at $(\tilde{h}(v_k), v_k)$. Recall that at each step of NlEm, a linear system is solved. Define the residual of this solve in the approximate case to be

$$r_k \equiv \tilde{g}'_k \begin{bmatrix} \Delta u_k \\ \Delta v_k \end{bmatrix} + \tilde{g}_k. \tag{5.3}$$

In this section, a bound is given for the α_k in (5.2). This bound will allow us to show that in the limit quadratic convergence is attainable if the g_1 system and (1.15) are solved accurately enough. The analysis of the convergence of approximate NlEm proceeds in two stages. In the first stage, an expression is derived that bounds the α_k in (5.2) in terms of the accuracy of solving the g_1 system of equations, i.e., $\|g_{1k}\|$. In the second stage, the terms of that expression are bounded in two ways with two theorems. The first theorem is existential, giving the conditions under which approximate NlEm attains higher-order convergence. These conditions are not verifiable. The second theorem gives computable conditions for approximately determining α_k.

5.1. Bounding α_k.

LEMMA 5.1. *Under the assumptions of Theorem* 4.2, *the* α_k *of* (5.2) *can be bounded by*

$$\begin{aligned} \alpha_k \|f_k\| = \alpha_k \|g_{2k}\| \le & \left\| \tilde{g}_{21k} \tilde{g}_{11k}^{-1} \int_0^1 \left[g_{11}(\tilde{h} + s(h - \tilde{h}), v_k) - \tilde{g}_{11k} \right] (h - \tilde{h}) ds \right\| \\ & + \left\| \int_0^1 \left[g_{21}(\tilde{h} + s(h - \tilde{h}), v_k) - \tilde{g}_{21k} \right] (h - \tilde{h}) ds \right\| \\ & + \|(g_{22k} - g_{21k} g_{11k}^{-1} g_{12k}) - (\tilde{g}_{22k} - \tilde{g}_{21k} \tilde{g}_{11k}^{-1} \tilde{g}_{12k})\| \; \|\Delta v_k\| \\ & + \|r_{2k} - \tilde{g}_{21k} \tilde{g}_{11k}^{-1} r_{1k}\|, \end{aligned} \tag{5.4}$$

where we have partitioned r_k *into* r_{1k}, r_{2k} *corresponding to the blocking of* g.

Proof. Recall that

$$\begin{aligned} \alpha_k \|f_k\| = \alpha_k \|g_{2k}\| & = \|(g_{21k} h' + g_{22k}) \Delta v_k + g_{2k}\| \\ & = \|(g_{22k} - g_{21k} g_{11k}^{-1} g_{12k}) \Delta v_k + g_{2k}\| \end{aligned} \tag{5.5}$$

where, from (5.3), Δv_k satisfies

$$\begin{bmatrix} \tilde{g}_{11k} & \tilde{g}_{12k} \\ \tilde{g}_{21k} & \tilde{g}_{22k} \end{bmatrix} \begin{bmatrix} \Delta u_k \\ \Delta v_k \end{bmatrix} + \begin{bmatrix} \tilde{g}_{1k} \\ \tilde{g}_{2k} \end{bmatrix} = \begin{bmatrix} r_{1k} \\ r_{2k} \end{bmatrix}. \tag{5.6}$$

That the $\tilde{g}_{ijk}$ exist, for $i, j = 1, 2$, and $\tilde{g}_{11k}$ nonsingular follows from

$$(\tilde{h}(v_k), v_k) \in S_0(g) \subset S_0(g_1). \tag{5.7}$$

From (5.6) it follows that

$$\begin{aligned} \Delta v_k = & (\tilde{g}_{22k} - \tilde{g}_{21k} \tilde{g}_{11k}^{-1} \tilde{g}_{12k})^{-1} (\tilde{g}_{2k} - \tilde{g}_{21k} \tilde{g}_{11k}^{-1} \tilde{g}_{1k}) \\ & + (\tilde{g}_{22k} - \tilde{g}_{21k} \tilde{g}_{11k}^{-1} \tilde{g}_{12k})^{-1} (r_{2k} - \tilde{g}_{21k} \tilde{g}_{11k}^{-1} r_{1k}). \end{aligned} \tag{5.8}$$

We begin by deriving an inequality relating g_{2k} in terms of $\tilde{g}_{2k}$. Applying the mean value theorem gives

$$\begin{aligned} g_{2k} &= \tilde{g}_{2k} + \int_0^1 g_2'(\tilde{h} + s(h-\tilde{h}), v_k)\begin{bmatrix} h-\tilde{h} \\ 0 \end{bmatrix} ds \\ &= \tilde{g}_{2k} + \tilde{g}_{21k}(h-\tilde{h}) + \int_0^1 \left[g_{21}(\tilde{h}+s(h-\tilde{h}), v_k) - \tilde{g}_{21k}\right](h-\tilde{h})ds. \end{aligned} \tag{5.9}$$

Similarly, for g_1,

$$\begin{aligned} 0 = g_{1k} &= \tilde{g}_{1k} + \tilde{g}_{11k}(h-\tilde{h}) \\ &\quad + \int_0^1 \left[g_{11}(\tilde{h}+s(h-\tilde{h}), v_k) - \tilde{g}_{11k}\right](h-\tilde{h})ds. \end{aligned} \tag{5.10}$$

We derive an expression for g_{2k} by solving for $h-\tilde{h}$ in the second term of (5.10) and substituting into (5.9). The expression is given by

$$\begin{aligned} g_{2k} &= \tilde{g}_{2k} - \tilde{g}_{21k}\tilde{g}_{11k}^{-1}\tilde{g}_{1k} \\ &\quad - \tilde{g}_{21k}\tilde{g}_{11k}^{-1}\int_0^1 \left[g_{11}(\tilde{h}+s(h-\tilde{h}), v_k) - \tilde{g}_{11k}\right](h-\tilde{h})ds \\ &\quad + \int_0^1 \left[g_{21}(\tilde{h}+s(h-\tilde{h}), v_k) - \tilde{g}_{21k}\right](h-\tilde{h})ds. \end{aligned} \tag{5.11}$$

Inequality (5.4) is determined by substituting (5.8) and (5.11) into (5.5) and applying the triangle inequality. □

PROPOSITION 5.2. *Under the assumptions of Theorem* 4.2, *the* α_k *of Lemma* 5.1 *satisfies*

$$\begin{aligned} \alpha_k\|g_{2k}\| &\le k_2(1+k_5k_3)^2c\|\tilde{g}_{1k}\|\|\Delta v_k\| \\ &\quad + \frac{k_3+k_5k_3^2}{2}c^2\|\tilde{g}_{1k}\|^2 + (1+k_3k_5)\|r_k\|, \end{aligned} \tag{5.12}$$

where $c = \frac{M}{1-\theta}$.

We preface the proof with a few remarks. Previously we noted the existence of $\tilde{g}_{ijk}$ and the nonsingularity of $\tilde{g}_{11k}$. The assumptions allow us to actually bound these quantities. That is, $\|\tilde{g}_{ijk}\| \le k_3, i, j = 1, 2$, and $\|\tilde{g}_{11k}^{-1}\| \le k_5$. Note also that

$$(\tilde{h}(v_k) + t_k\Delta u_k, v_k + t_k\Delta v_k) \in S_1(g) \subset S_0(g_1). \tag{5.13}$$

Thus we may use Δu_k to generate a starting guess for the solution of $g_1(u, v_k + t_k\Delta v_k)$. The algorithmic implication of this point is that in the regime of quadratic convergence of (the whole) g, NlEm need not do a solve of the g_1 equations; they will already be small enough. Finally, note that

$$(h(v_k), v_k), (\tilde{h}(v_k), v_k) \in S_0(g_1); \tag{5.14}$$

thus

$$(\tilde{h}(v_k) + s(h(v_k) - \tilde{h}(v_k)), v_k) \in S_1(g_1) \tag{5.15}$$

for $0 \le s \le 1$ and therefore the Lipschitz condition holds along the line segment.

Proof. The proof proceeds by bounding each of the terms of (5.4). Expanding norms and applying the Lipschitz condition to the first term in (5.4) yields

$$\|\tilde g_{21k}\tilde g_{11k}^{-1}\int_0^1\Big[g_{11}(\tilde h+s(h-\tilde h),v_k)-\tilde g_{11k}\Big](h-\tilde h)ds\| \le k_5\frac{k_3^2}{2}\|h-\tilde h\|^2. \tag{5.16}$$

Similarly, the second term can also be bounded yielding

$$\left\|\int_0^1\Big[g_{21}(\tilde h+s(h-\tilde h),v_k)-\tilde g_{21k}\Big](h-\tilde h)ds\right\| \le \frac{k_3}{2}\|h-\tilde h\|^2. \tag{5.17}$$

Applying Proposition 3.7 and Property P1 to the third term of (5.4), we get

$$\|(g_{22k}-g_{21k}g_{11k}^{-1}g_{12k})-(\tilde g_{22k}-\tilde g_{21k}\tilde g_{11k}^{-1}\tilde g_{12k})\| \le k_2(1+k_5k_3)^2\|h-\tilde h\|. \tag{5.18}$$

The fourth term is bounded by applying Properties P1 and P2 to note that

$$\|r_{2k}-\tilde g_{21k}\tilde g_{11k}^{-1}r_{1k}\| \le \|L^{-1}\|\|r_k\|, \tag{5.19}$$

where

$$L=\begin{bmatrix} I & 0\\ \tilde g_{21k}\tilde g_{11k}^{-1} & I\end{bmatrix}. \tag{5.20}$$

Thus

$$\|r_{2k}-\tilde g_{21k}\tilde g_{11k}^{-1}r_{1k}\| \le (1+k_3k_5)\|r_k\|. \tag{5.21}$$

Collecting (5.16) - (5.21) gives

$$\begin{aligned}\alpha_k\|g_{2k}\| &\le k_2(1+k_5k_3)^2\|h-\tilde h\|\|\Delta v_k\|\\ &\quad+\frac{k_3+k_5k_3^2}{2}\|h-\tilde h\|^2+(1+k_3k_5)\|r_k\|.\end{aligned} \tag{5.22}$$

Since $h(v_k)$ is the root of the equation $g_1(u,v_k)$ with v_k fixed and the assumptions assure that GAN converges from $\tilde h(v_k)$, we may apply Proposition 3.3 to conclude that

$$\|h-\tilde h\| \le c\|\tilde g_{1k}\|. \tag{5.23}$$

Substituting (5.23) into (5.22) gives the desired result. □

5.2. Convergence theorems. We continue by giving conditions for bounding the terms of α_k in (5.12). Two theorems are presented. The first theorem gives the conditions under which higher-order convergence can be expected; however, the result is existential. It is not possible to verify the conditions computationally. The second theorem gives a method for testing the size of α_k computationally.

THEOREM 5.3. *Under the assumptions of Proposition* 5.2, *if conditions*
C1: $\|g_1(\tilde h(v_k),v_k)\| \le \|g_2(h(v_k),v_k)\|$, *and*
C2: $\|r_k\| \le \|g_2(h(v_k),v_k)\|^{(1+p)}$
hold, then

$$\begin{aligned}\alpha_k\|g_{2k}\| &\le (1+k_3k_5)\|g_{2k}\|^{1+p}\\ &\quad+\left(2k_1k_2(1+k_5k_3)^2c_1+\frac{k_3+k_5k_3^2}{2}c_1\right)\|\tilde g_{2k}\|^2,\end{aligned} \tag{5.24}$$

where

$$\|\tilde{g}_{2k}\| \le (1 + k_5 c_1)\|g_{2k}\| + \frac{k_3}{2} c_1^2 \|g_{2k}\|^2. \tag{5.25}$$

Proof. The theorem follows from Proposition 5.2 by bounding the quantities $\|\tilde{g}_{1k}\|$, $\|\Delta v_k\|$, and $\|\tilde{g}_{2k}\|$. The bound for $\|\tilde{g}_{1k}\|$ is given by condition C1.

By Assumption A2 we know that

$$\left\| \begin{bmatrix} \Delta u_k \\ \Delta v_k \end{bmatrix} \right\| \le k_1 \|\tilde{g}_k\|. \tag{5.26}$$

Applying Properties P1 and P2 and condition C1 it follows that

$$\|\Delta v_k\| \le k_1(\|\tilde{g}_{1k}\| + \|\tilde{g}_{2k}\|) \le 2k_1 \|\tilde{g}_{2k}\|. \tag{5.27}$$

All that remains is to bound $\|\tilde{g}_{2k}\|$. We apply the mean value theorem to give

$$\tilde{g}_{2k} = g_{2k} + \int_0^1 g_2'(h + s(\tilde{h} - h), v_k) \begin{bmatrix} \tilde{h} - h \\ 0 \end{bmatrix} ds. \tag{5.28}$$

Following the methodology that led to inequality (5.17) and applying the triangle inequality and (5.23) gives

$$\|\tilde{g}_{2k}\| \le \|g_{2k}\| + k_5 c_1 \|\tilde{g}_{1k}\| + \frac{k_3}{2} c_1^2 \|\tilde{g}_{1k}\|. \tag{5.29}$$

Finally, applying condition C1 yields the inequality (5.25). □

Neither condition C1 nor C2 are verifiable. The following theorem gives conditions that are verifiable.

THEOREM 5.4. *Under the assumptions of Proposition* 5.2, *if conditions*

C1a: *the g_1 equation is solved for $\tilde{h}(v_k)$ so that $\|g_1(\tilde{h}(v_k), v_k)\| \le \|g_2(\tilde{h}(v_k), v_k)\|$, and*

C2a: $\|r_k\| \le \|g_2(\tilde{h}(v_k), v_k)\|^{(1+p)}$

hold, then

$$\begin{aligned} \alpha_k \|g_{2k}\| &\le (1 + k_3 k_5)\|\tilde{g}_{2k}\|^{1+p} \\ &\quad + \left(2k_1 k_2 (1 + k_5 k_3)^2 c_1 + \frac{k_3 + k_5 k_3^2}{2} c_1\right) \|\tilde{g}_{2k}\|^2. \end{aligned} \tag{5.30}$$

The proof of Theorem 5.4 follows directly from the proof of Theorem 5.3. The intention of Theorem 5.4 is twofold. First, if $\|\tilde{g}_{2k}\|$ is small enough, then at the very least $\alpha_k < 1$ and, therefore, the method is converging (cf. Proposition 3.1). Second, in the limit when $\|r_k\| \le \|\tilde{g}_{2k}\|^2$, quadratic convergence is attained.

Note that in the statement of Theorem 5.4 we write that g_1 should be solved so that $\|g_1(\tilde{h}(v_k), v_k)\| < \|g_2(\tilde{h}(v_k), v_k)\|$. Unfortunately, this inequality can only be checked after a function evaluation is done to determine $g_2(\tilde{h}(v_k), v_k)$. To avoid having to recompute g_2 every time a new $\tilde{h}(v_k)$ is generated, i.e., after every iteration of GAN on the g_1 system, we suggest that g_1 be solved so that $\|g_1(\tilde{h}(v_k), v_k)\| < \|g_2(\tilde{h}(v_{k-1}), v_{k-1})\|^2$.

COROLLARY 5.5. *Under the conditions of Theorem* 5.4, *with $p = 1$, if* C1a *is replaced with*

$$\|g_1(\tilde{h}(v_k), v_k)\| = O(\|g_2(\tilde{h}(v_{k-1}), v_{k-1})\|^2), \tag{5.31}$$

then when $\|g_2(\tilde{h}(v_{k-1}), v_{k-1})\|$ *is sufficiently small, i.e., close enough to the root, the convergence of the method is quadratic.*

Proof. The proof follows from Theorem 3.2 with the observation that sufficiently close to the root

$$\|g_2(\tilde{h}(v_k), v_k)\| = O(\|g_2(\tilde{h}(v_{k-1}), v_{k-1})\|^2), \tag{5.32}$$

and therefore condition C1 holds. □

5.3. Accuracy requirement for g_1 system. As pointed out in §2, inaccuracies in computing f_k cause a practical problem in the solution process. Even if the seemingly fortuitous inequality

$$\|g_2(\tilde{h}(v_k), v_k)\| < \|f(v_k)\| \tag{5.33}$$

holds, there might not be any t_k such that

$$\|g_2(\tilde{h}(v_{k+1}), v_{k+1})\| < \|g_2(\tilde{h}(v_k), v_k)\|. \tag{5.34}$$

Algorithmically, failure to satisfy equation (5.34) for a given choice of t_k leads to two options. The first is to keep trying smaller t_k's. The second is to go back to the previous iteration because the g_1 system was not solved accurately at that point. This section gives an approximate formula for determining whether $\|\tilde{g}_{1k}\|$ is close enough to zero to make $\|\tilde{g}_{2k+1}\| < \|\tilde{g}_{2k}\|$ a possibility. Satisfying this formula will give the user confidence that a norm reduction will eventually occur for t_k small enough.

PROPOSITION 5.6. *Recall* $f_k \equiv g_2(h(v_k), v_k)$. *Define* $\tilde{f}_k$ *to be an approximation of* f_k. *Let*

$$\|\tilde{f}_k - f_k\| \le \delta_k \|f_k\|. \tag{5.35}$$

If $0 < \delta_k,\ \delta_{k+1} < 1$, *and* $\|\tilde{f}_{k+1}\| \le \mu_1 \|\tilde{f}_k\|$, $\mu_1 < 1$, *then*

$$\|f_{k+1}\| \le \mu_2 \|f_k\|, \tag{5.36}$$

where

$$\mu_2 \|f_k\| \equiv \mu_1 \frac{(1+\delta_k)}{1-\delta_{k+1}} \|f_k\|. \tag{5.37}$$

Proof. We have

$$\begin{aligned} \|\tilde{f}_{k+1}\| &\le \mu_1 \|\tilde{f}_k\| \\ &\le \mu_1 \|\tilde{f}_k - f_k\| + \mu_1 \|f_k\| \\ &\le \mu_1 \delta_k \|f_k\| + \mu_1 \|f_k\| = (1+\delta_k)\mu_1 \|f_k\|; \end{aligned} \tag{5.38}$$

adding $\|\tilde{f}_{k+1} - f_{k+1}\|$ to both sides gives

$$\begin{aligned} \|\tilde{f}_{k+1} - f_{k+1}\| + \|\tilde{f}_{k+1}\| &\le (1+\delta_k)\mu_1 \|f_k\| + \|\tilde{f}_{k+1} - f_{k+1}\| \\ \|f_{k+1}\| &\le (1+\delta_k)\mu_1 \|f_k\| + \delta_{k+1} \|f_{k+1}\|; \end{aligned} \tag{5.39}$$

solving for $\|f_{k+1}\|$ gives

$$\|f_{k+1}\| \le \mu_1 \frac{(1+\delta_k)}{1-\delta_{k+1}} \|f_k\|. \quad \square \tag{5.40}$$

```
**NlEm** $(l, w^l, tol, MAX)$ /* at the outer level, l = 1 */
    **if** $g^l$ consists of no equations, **return**.
    **NlEm**$(l+1, P^{l+1}(w^l), tol_{-1}, MAX_{-1})$
    Note that this call to NlEm changes the $w^{l+1}$ unknowns.
    $k = 0, w_0^l = w_l$
    **while** $((\|g^l(w_k^l)\| > tol)$ **and** $(k < MAX))$
        compute $x_k^l$
        **repeat**
            choose a $t_k$
            $\hat{w}_{k+1}^l = w_k^l + t_k x_k^l$
            **NlEm**$(l+1, P^{l+1}(\hat{w}_{k+1}^l), tol_k, MAX_k)$
        **until** $(\|g^l(\hat{w}_{k+1}^l)\| < \theta \|g^l(w_k^l)\|)$ **or** $t_k$ is too small
        **if** $t_k$ is too small, **return** with $w^l$ unchanged
        $k = k+1, w_k^l = \hat{w}_k^l$
    **if** $(k > max)$ **return** with $w^l$ unchanged
    **return** with $w^l = w_k^l$
```

FIG. 5. *A detailed view of the NlEm algorithm.*

For our purposes we will assume that there is some μ_2 we wish to achieve. We compute a candidate $\tilde{f}_{k+1}$, yielding a μ_1. We are then able to determine if μ_2 is achievable. Recall from (5.11) and subsequent analysis that

$$\|\tilde{g}_{2k} - g_{2k}\| \le C\|h - \tilde{h}\|^2 + \|\tilde{g}_{21k}\tilde{g}_{11k}^{-1}\tilde{g}_{1k}\|. \tag{5.41}$$

An estimate for δ_k can be obtained by dropping the $C\|h - \tilde{h}\|^2$ term. Hence

$$\delta_k \cong \frac{\|\tilde{g}_{21k}\tilde{g}_{11k}^{-1}\tilde{g}_{1k}\|}{\|\tilde{g}_{2k} - \tilde{g}_{21k}\tilde{g}_{11k}^{-1}\tilde{g}_{1k}\|}, \tag{5.42}$$

where the denominator is an $O(\|h - \tilde{h}\|^2)$ approximation for $g_2(h(v_k), v_k)$.

6. Implementation details. While the outline of NlEm given in §2 is sufficient for a programmer to generate a piece of code, we have noticed certain improvements that make the transition from NlEm back to GAN transparent. Thus, in this section we present a more detailed algorithm and address certain issues raised in §5.

The algorithm in Figure 5 is a recursive algorithm that allows NlEm to be performed on more than two levels; i.e., NlEm is used to solve the g_1 equations. One place we view this as a possibility is on a system of nonlinear PDEs. The g_1 equation at one level might be one or more of the PDEs, while the g_1 equation at the next level might be the equations associated with some of the grid points. While we believe this to be a case where nested NlEM might be useful, we have not actually implemented NlEm for this problem.

Define by g^l the set of equations associated with level l. g^1 is the full set of equations. Define P^{l+1} to be a projector function that, when applied to the variables w^l (the variables associated with level l), returns w^{l+1}.

There are several interesting points about this routine.

1. If g^2 consists of no equations then the code above is GAN.
2. Notice that after computing x_k^l, all of $\hat{u}_{k+1}^l$ is updated. When the answer is close enough to the root that GAN is in the regime of quadratic convergence, the call

to NlEm at the next step will (if the tol_k is chosen reasonably) return doing no computation. Thus, the transition from NlEm back to GAN is achieved with just the cost of an extra function call.

3. We make tol_k smaller and MAX_k bigger as t_k decreases. As pointed out in §5 there is a problem with determining whether norm reduction fails because t_k has not been made small enough, or if the g_1 equations were not solved accurately enough at the previous level. While solving the g_1 equations more accurately at this level does not help the problem of failure at the moment, we have more confidence that it will not be a problem on the next iteration.
4. The maximum number of NlEm iterations needs to be set carefully. MAX_{-1} should be set very high so that NlEm does not give up too early trying to solve them. On the other hand MAX_k should be set to a small level (increasing as t_k decreases). Long periods of time should not be spent trying to get solutions for t_k's when some smaller t_k will get an answer faster. We know that the answer can be found for some smaller t_k since the u from the previous iteration can be used if t_k is sufficiently close to 0.
5. It is sometimes the case that g^2 is not known before beginning. In that case g^2 would be calculated before the initial call to **NlEm**. It is important that g^2 not be changed after this. If for some reason the user does wish to dynamically change g^2, it is important to check that after the initial call to **NlEm** there has been a norm reduction. If this is not done, thrashing can occur and the code will not terminate.

7. Examples. In this section we give examples showing the performance of NlEm. The first two examples demonstrate how NlEm is used for a system of nonlinear equations arising from the discretization of an ODE. The first example shows that NlEm can improve performance on a problem where grid refinement might otherwise have been used. That is, the user would recognize that most of the action is taking place at some set of points and the rest of the domain is quiescent. Instead of gridding the domain differently, a fine uniform grid is used, and more computation is done on the points of interest. The cost of the algorithm is improved by five times. The second example shows how NlEm can be used to solve problems where too few grid points are used. When too few grid points are used, the steep valleys as seen in §2 sometimes occur. NlEm can be used on the "bad" grid points giving results where GAN fails. The third example is a simple 2-D semiconductor device. By eliminating two of the PDEs, we get convergence with an example that would not converge using GAN.

7.1. Elimination by grid 1. Consider the nonlinear boundary value problem given by

$$\begin{aligned} &-u'' + u^3 + \left(4\frac{(x-0.5)^2}{10^8} - 2\times 10^{-4}\right)u - 10^9 e^{-3(\frac{x-0.5}{0.01})^2} = 0, \\ &u(0) = 0, \quad u(1) = 0. \end{aligned} \tag{7.1}$$

This has the solution

$$u(x) = 10^3 e^{-(\frac{x-0.5}{0.01})^2}. \tag{7.2}$$

The action in this problem is around $x = 0.5$, where a large spike occurs.

We give a few details about this example using 100 discretized equations. An initial guess for $u_1 \ldots u_{100}$ is made. The g_1 system is the discretized equations 49-52. Using the initial guess for u_{48} and u_{53} as fixed values, the nonlinear subsystem g_1 is solved to a very high accuracy (e.g., 10^{-12} for the norm of the residual). Following this initial solve of the g_1 system, the main loop begins. The entire system g is evaluated at the new point; that is, $u_1 - u_{48}$ and $u_{53} - u_{100}$ are unchanged and $u_{49} - u_{52}$ have the values just computed. The

TABLE 1

Comparison of NlEm to GAN for example given in §7.1. *"Points" refers to the number of grid points in the domain. "its" refers to the number of iterations required. The grid points between LEFT and RIGHT are in* g_1.

Points	GAN time	GAN its	NlEm time	NlEm its	LEFT	RIGHT
100	0.15	10	0.03	5	48	53
500	1.47	10	0.17	5	240	261
1000	2.81	10	0.32	5	480	521
5000	10.7	10	1.66	5	2400	2601

Jacobian is computed and the Newton equations are solved, giving the update vector Δu. Then a damping parameter t is chosen, and $u_{48} + t\Delta u_{48}$ and $u_{53} + t\Delta u_{53}$ are computed. These are used as fixed values in the g_1 solve, as above, except that the accuracy of the solution is relaxed, to say $t * (\|g\|/\|g_i\|)^2$, where g_i is the residual at the initial guess. Although we used the damping parameter to control the accuracy, any function of t can be used that shows similar behavior; i.e., the accuracy requirement is increased for smaller t and is relaxed when t approaches 1. This ensures that as the regime of quadratic convergence is approached, few (if any) iterations of Newton's method on equations 49-52 are required. The main loop repeats when a damping parameter is chosen so that a norm reduction for the whole g has occurred following the g_1 solve.

In Table 1 we see that NlEm runs about five times faster than GAN. It is so much faster because in the first few iterations of GAN, very small values of the damping parameter are required. The very small damping parameters mean that the function must be evaluated many times. NlEm also has to use small values for the damping parameter for the g_1 equations, but performing a large number of function evaluations on four percent of the points is not very costly.

7.2. Elimination by grid 2. Consider the nonlinear boundary value problem given by

$$\begin{gathered} -u'' + \alpha(x)u' + u^3e^u + k(x) = 0, \\ u(0) = 0, \quad u(1) = 0, \\ \alpha(x) = 10^9 e^{-(\frac{x-0.5}{0.01})^2}\sqrt{|x-0.5|}, \quad k(x) = \begin{cases} -10^6 & x < 0.5, \\ 10^6 & x \geq 0.5, \end{cases} \end{gathered} \tag{7.3}$$

where u' is discretized using central differences. Fifty grid points were used on this problem; but because central differences were used, this was not enough grid points. Using the same code as in the previous example, the computed solution under these circumstances shows ringing around 0.5. GAN is unable to solve this problem. NlEm solves the problem in 17 iterations. Both methods solve the problem when more grid points are used.

7.3. Example from semiconductor device simulation. One of the areas in which we have been directing the application of this method is semiconductor device simulation [9]. NlEm was implemented in the semiconductor device simulator SIMUL [11]. In the example presented here, the equations take the form

$$-\nabla \cdot (\epsilon \nabla u) + n - p + N = 0, \tag{7.4}$$

$$-\nabla \cdot \mathbf{J}_n + R_n(n, p) = 0, \tag{7.5}$$

$$-\nabla \cdot \mathbf{J}_p + R_p(n, p) = 0, \tag{7.6}$$

where u, n, and p are functions in space (for the example we present here, $(x, y) \in \mathbb{R}^2$), ϵ and N are spatially related constants, J_n and J_p are electron and hole current densities, and R_n and R_p are recombination terms.

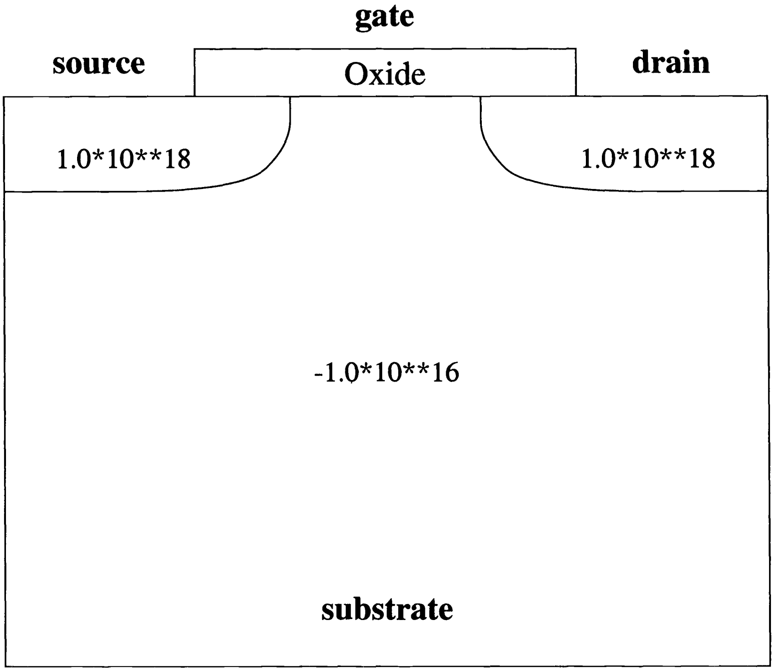


FIG. 6. *NMOS transistor. The numbers represent the background dopings of the N and P regions.*

These steady state equations are usually solved using GAN. Under certain circumstances GAN fails to converge or converges too slowly. In these cases, other methods are employed to generate a good initial guess so that GAN converges well. These methods include nonlinear Gauss-Seidel and continuation in the boundary conditions. In nonlinear Gauss-Seidel each equation is solved in turn using the most recent values for the other variables as constants. Sometimes nonlinear Gauss-Seidel also fails in which case continuation in the voltage at the contacts is employed.

The example given in Figure 6 is an NMOS transistor. The voltages at the various contacts are source=0, gate=1, substrate=0, and drain=1 volts. The problem was originally solved using a combination of nonlinear Gauss-Seidel and continuation in 235 seconds. When the system is solved with NlEm, with the g_1 equations being the electron and hole continuity equations, the solution time is 36 seconds. Other choices for the g_1 equations did not perform as well. In the experience of the author of SIMUL, choosing electron and hole continuity equations to be g_1 works in many cases. We also point out that when the problem is easy, e.g., for drain voltage of 0.1 volts, NlEm takes longer than GAN. We reiterate that NlEm should be used in cases where GAN is having difficulties.

Acknowledgment. We are indebted to Stephan Müller for implementing NlEm in the semiconductor device simulation package SIMUL and for patiently helping the authors find an interesting example. We also thank Wolfgang Fichtner for allowing us to spend time with his student.

REFERENCES

[1] R. E. BANK AND D. J. ROSE, *Global approximate Newton methods*, Numer. Math., 37 (1981), pp. 279-295.

[2] R. E. BANK, D. J. ROSE, AND W. FICHTNER, *Numerical methods for semiconductor device simulation*, SIAM J. Sci. Statist. Comput., 4 (1993), pp. 416-435.

[3] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*, Academic Press, New York, 1982.

[4] K. M. Brown, *A quadratically convergent Newton-like method based upon Gaussian elimination*, SIAM J. Numer. Anal., 6 (1969), pp. 560-569.

[5] W. M. Coughran, E. Grosse, and D. J. Rose, *CAzM: A circuit analyzer with macromodeling*, IEEE Trans. Computer-Aided Design, ED-30, (1983), pp. 1207-1213.

[6] ———, *Aspects of computational circuit analysis*, in VLSI CAD Tools and Applications, W. Fichtner and M. Morf, eds, Kluwer Academic Publishers, Norwell, MA, 1987.

[7] G. H. Golub and V. Pereyra, *The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate*, SIAM J. Numer. Anal., 10 (1983), pp. 413-432.

[8] ———, *Differentiation of pseudo-inverses, separable nonlinear least squares problems and other tales*, in Generalized Inverses and Applications, M. Z. Nashed, ed., Academic Press, New York, 1976, pp. 303-324.

[9] P. J. Lanzkron, D. J. Rose, J. T. Wilkes, S. Müller, and W. Fichtner, *The use of nonlinear elimination in steady-state circuit and device simulation*, Proc. of the NUPAD IV, Seattle, WA, IEEE, 1992.

[10] W. H. Lawton and E. A. Sylvestre, *Elimination of linear parameters in nonlinear regression*, Technometrics, 13 (1971), pp. 461-467.

[11] S. Müller, K. Kells, J. Litsios, U. Krumbein, A. Schenk, and W. Fichtner, *SIMUL* 1.1 *Manual*, Integrated Systems Laboratory, ETH Zurich, Switzerland, 1993.

[12] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.

[13] J. T. Wilkes, *A New Method for Solving Systems of Nonlinear Equations in Circuit Simulation*, Ph.D. thesis, Duke University, Durham, NC, 1994.

SIAM J. SCI. COMPUT.
Vol. 17, No. 3, pp. 561-578, May 1996

THREE-DIMENSIONAL STEADY FLOW IN A DIVIDING CHANNEL USING FINITE AND PSEUDOSPECTRAL DIFFERENCES*

ROLAND HUNT†

Abstract. A numerical solution for the flow of a steady incompressible fluid in a general symmetric three-dimensional bifurcation having uniform thickness is presented for Reynolds numbers up to 100. The physical region is divided into three subdomains, each of which is transformed onto a rectangular parallelepiped upon which the calculations are made. The Navier–Stokes equations are differenced using standard second-order differences in the direction of each tube and pseudospectral differences in the two transverse directions. The use of such differences in the transverse plane greatly reduces the number of nodes required for a given accuracy, and this particular mix of differences minimizes the CPU time requirement when Newton's method is used to solve the resulting algebraic system. Two bifurcations have been considered with daughter tubes angled to the main tube at 60° and 90°, respectively. By comparing results on different sized grids the estimated relative accuracy is typically of order 10^{-3}. The resulting flows display novel three-dimensional features not present in two-dimensional flows. In particular, as fluid passes from main to daughter tubes it twists in a fairly complex vortex-like fashion.

Key words. pseudospectral methods, three-dimensional flow

AMS subject classification. 65N35

1. Introduction.

In a previous paper [11] we presented the numerical solution of the steady incompressible flow in a three-dimensional tube having general specifiable boundaries using a judicious combination of finite and pseudospectral differences. The ensuing nonlinear system was solved using Newton's iteration. This blend of finite and pseudospectral differences meant that it was possible to obtain results of reasonable accuracy on a workstation.

The aim of that paper was first to examine the possibility of using such a mix of finite and pseudospectral differences. Along the tube second-order central finite differences are employed but in the two transverse directions pseudospectral differences are used (see [7] for an overview). For a given tolerance this particular mix of differences is optimal in the sense that it gives a minimum CPU requirement when Newton's method is employed to solve the resulting algebraic system which was the method adopted. The use of pseudospectral differences across the tube, rather than finite differences, greatly reduces the number of nodes required for a given accuracy which, in turn, greatly reduces the CPU requirement. Typically for an accuracy of 10^{-3}, the number of nodes required across the tube is 10 compared with 50 for finite differences which leads to an enormous saving of storage (typically 500) and CPU time (typically 10000). However, replacing finite differences by pseudospectral differences along the tube can considerably increase the CPU requirement. It was shown that using pseudospectral discretizations in all three directions can result in the CPU time being increased by as much as a factor of 60. This is because the Jacobian matrix employed in Newton's method, which is banded when pseudospectral differences are only used across the tube, now becomes full.

A second aim was the use and suitability of Newton's method for solving the algebraic system resulting from a three-dimensional flow problem. The use of Newton's iteration to provide a numerical solution for fluid flow problems was pioneered by Fornberg [3]-[6] and studied by the author [9], [10]. The advantage of using Newton's method is that it is second-order convergent. This means that provided a suitable starting iterate is available, which is not a difficulty for flow problems of this type, the iteration always converges and, furthermore, only requires three or four iterations for a tolerance of $\sim 10^{-6}$. This contrasts with traditional SOR methods in which, for large Reynolds number, some kind of additional artificial viscosity

*Received by the editors July 23, 1994; accepted for publication (in revised form) February 16, 1995.

†Department of Mathematics, University of Strathclyde, Glasgow, Scotland (r.hunt@uk.ac.strath).

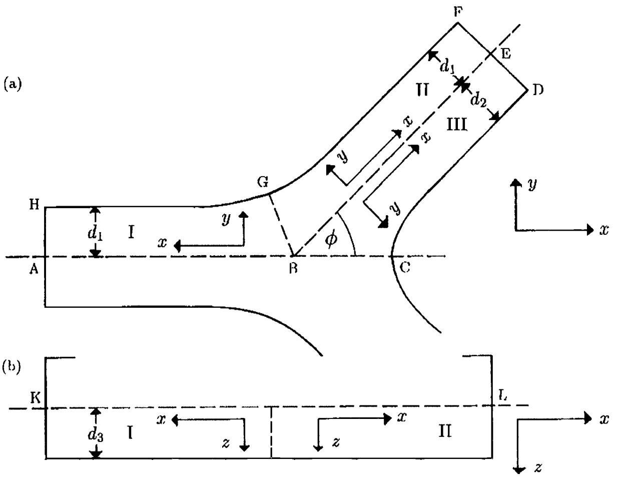


FIG. 1. *The geometry of the bifurcation showing* (a) *side view and* (b) *plan. The bifurcation is symmetric about AC and KL. The coordinates used for the three domains ABGH, BEFG, and BCDE are shown inside each domain. The physical coordinates are shown outside the domain.*

or upwinding is necessary, the introduction of which can seriously degrade the accuracy of the solution [4], [9]. The cost of each iteration in terms of CPU time is high but, because of the few iterations required, the total time is comparable with that needed by an SOR method for a two-dimensional flow and somewhat higher for three dimensions. A disadvantage of the method is that the Jacobian is so huge that it needs to be stored outside the memory of the computer; for example, on a hard disk. However, by transferring data in large batches the program can be made very efficient.

This study of the flow in a tube was a precursor to the study of the flow in a three-dimensional bifurcation in which three tubes are patched together. This is the subject of this paper. The motivation for such a study is the simulation of blood flow through an arterial bifurcation which is known to be a preferred site for atherosclerosis (see [12] for a review). The vast majority of numerical fluid flow calculations have two space dimensions and little has been done in three dimensions. A physical arterial bifurcation is three dimensional and it is known that the flow contains significant secondary circulations not present in a two-dimensional representation and hence the need for a full three-dimensional numerical study of the problem.

In this paper we present the numerical calculation of the three-dimensional, steady, incompressible flow in a symmetric bifurcation having constant thickness (Fig. 1). The bifurcation is divided into three domains labeled I, II, and III, and the flow in each domain is solved numerically in a similar manner as for the generalized tube. First each domain is mapped onto a rectangular parallelepiped having a rectilinear grid on which the governing equations are solved. Because the mapping is quite general the boundaries CD and FH can be specified arbitrarily. Second, as for the generalized tube, the derivatives of the governing equations are approximated by second-order central differences along the tube and by pseudospectral differences in the two transverse directions. And, finally, the resulting nonlinear system of equations is solved using Newton's method.

2. The governing equations, their transformation and differencing. The equations governing the flow of an incompressible fluid are given by the Navier–Stokes equations. In a steady state these are

(2.1a) $$\nabla.\mathbf{u} = 0,$$

(2.1b) $$(\mathbf{u}.\nabla)\mathbf{u} + \frac{1}{\rho}\nabla p = \nu\nabla^2\mathbf{u},$$

where (2.1a) is the continuity constraint and (2.1b) the momentum balance. The symbols ρ, $\mathbf{u}$, p, and ν are the density, velocity, pressure, and kinematic velocity, respectively. If U and L are representative of the central velocity and typical "width" of a tube then we nondimensionalize the equations using

(2.2) $$\mathbf{u} = U\mathbf{u}' \qquad p = \frac{\rho U^2}{R}p'$$

to give

(2.3a) $$\nabla.\mathbf{u} = 0,$$

(2.3b) $$R(\mathbf{u}.\nabla)\mathbf{u} + \nabla p = \nabla^2\mathbf{u},$$

where the dashes have been dropped for clarity and $R = UL/\nu$ is the Reynolds number. The pressure has been nondimensionalized in this manner so that setting $R = 0$ in (2.3b) gives the Stokes equation.

To solve equations (2.3), each domain (I, II, or III) is first transformed onto a rectangular parallelepiped such that the original boundary is transformed onto a boundary of the parallelepiped. We will use (x, y, z) or (x^1, x^2, x^3), as shown in Fig. 1, to denote the original coordinates and (ξ, η, ζ) or (ξ^1, ξ^2, ξ^3) as the coordinates in transform space. Similarly the velocity components in the (x, y, z) directions will be denoted by (u, v, w) or (u^1, u^2, u^3). Then equations (2.3) are

(2.4a) $$\frac{\partial u^J}{\partial x^J} = 0,$$

(2.4b) $$Ru^J\frac{\partial u^I}{\partial x^J} + \frac{\partial p}{\partial x^I} = \frac{\partial^2 u^I}{\partial x^J \partial x^J},$$

where $I = 1, 2, 3$ and repeated superscripts are summed from 1 to 3. The transformed equations then are

(2.5a) $$\frac{\partial \xi^K}{\partial x^J}\frac{\partial u^J}{\partial \xi^K} = 0,$$

(2.5b) $$Ru^J\frac{\partial \xi^K}{\partial x^J}\frac{\partial u^I}{\partial \xi^K} + \frac{\partial \xi^K}{\partial x^I}\frac{\partial p}{\partial \xi^K} = \frac{\partial \xi^K}{\partial x^J}\frac{\partial \xi^L}{\partial x^J}\frac{\partial^2 u^I}{\partial \xi^K \partial \xi^L} + \frac{\partial^2 \xi^K}{\partial x^J \partial x^J}\frac{\partial u^I}{\partial \xi^K}.$$

The transformations used in this paper are expressed algebraically and the terms $\partial\xi^K/\partial x^J$ and $\partial^2\xi^K/\partial x^J\partial x^J$ were determined using the symbolic algebraic package Maple [2]. The terms $\partial u^I/\partial x^K$ and $\partial^2 u^I/\partial x^K \partial x^K$ are approximated using difference formulae.

The grid employed in transform space is shown in Fig. 2. In the ξ direction the spacing $\Delta\xi$ is uniform and the transformation is chosen such that $\Delta\xi = 1$. We will assume, without loss of generality, that η has domain $0 \le \eta \le 1$ and choose the Gauss-Lobatto collocation points as nodal values; that is

(2.6) $$\eta_j = \frac{1}{2}\left(1 - \cos\frac{\pi j}{n}\right), \qquad j = 0, 1, \ldots, n$$

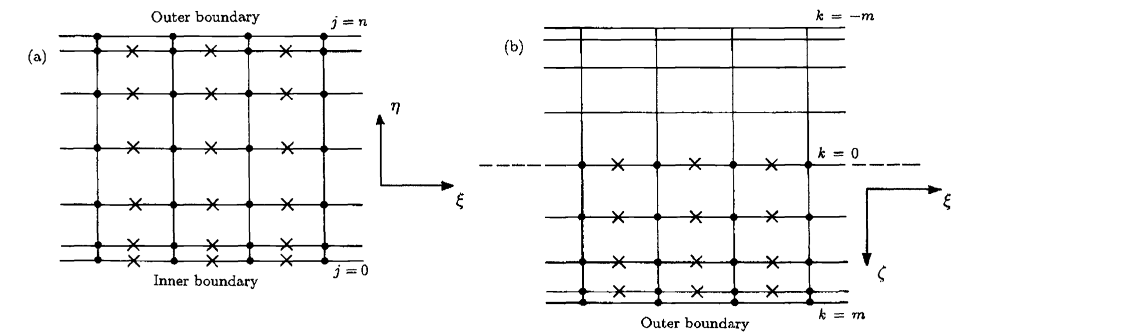


FIG. 2. *Sections of the grid in* (a) *the ξ-η planes and* (b) *the ξ-ζ planes showing the position of the velocity nodes (filled circles) and pressure nodes (crosses). In* (b) *the dashed line shows the axis of symmetry.*

for a specified integer n. Similarly the domain for ζ is $0 \le \zeta \le 1$ but, in order to take full advantage of the symmetry about $\zeta = 0$, we will use the Gauss-Lobatto points on $-1 \le \zeta \le 1$; namely,

$$\zeta_k = \sin\frac{\pi k}{m}, \qquad k = 0, 1, \ldots, m \tag{2.7}$$

for which we regard $k = -m, -m+1, \ldots, -1$ as auxiliary fictitious nodes. These will enable us to have difference formulae of optimum accuracy in the ζ direction.

On the grid the velocity components u^I are associated with integer points (i, j, k), $0 \le i \le l + 1$, $0 \le j \le n$, $0 \le k \le m$, and the pressure p with points $(i + \frac{1}{2}, j, k)$, $0 \le i \le l$, $1 \le j \le n - 1$, $1 \le k \le m - 1$ where i denotes the integer collocation points in the ξ direction. For fluid flows there is no physical boundary condition for the pressure similar to the no-slip condition for the velocity, although one can be derived from the Navier–Stokes equations. This poses considerable numerical difficulties, first, in setting up a complete set of algebraic equations and, second, in obtaining a stable solution due to the presence of several spurious pressure modes [1]. Both of these problems can be overcome by simply not using the pressure collocation points on the boundary ($j = n$ or $k = m$) and using an interpolant of one degree less in the η and ζ directions. This avoids the need to supply a pressure boundary condition and there will be no spurious pressure modes since there no longer exists a nonconstant pressure solution which vanishes at all interior collocation points. In fact an attempt to include boundary pressure points resulted in a highly oscillatory saw-tooth solution which clearly shows the presence of unstable modes. Another possibility which would have a similar effect would be to use a staggered grid for the pressure similar to that originally proposed by Harlow and Welch in their "marker and cell" method [8]. This would be more difficult to program without any obvious advantage. However, we use half-integer points for the pressure in the i-direction since this gives a more accurate difference formula for the first derivative in ξ direction.

The derivatives of u^I in the ξ direction are approximated by second-order central differences which about the point (i, j, k) are

$$\frac{\partial u^I}{\partial \xi} \simeq G_1^I \equiv \frac{1}{2}\left(u^I_{i+1,j,k} - u^I_{i-1,j,k}\right), \quad \frac{\partial^2 u^I}{\partial \xi^2} \simeq H^I_{1,1} \equiv u^I_{i+1,j,k} - 2u^I_{i,j,k} + u^I_{i-1,j,k}, \tag{2.8}$$

and in the η direction by pseudospectral differences given by

$$\frac{\partial u^I}{\partial \eta} \simeq G_2^I \equiv \sum_{jj=0}^{n} D^n_{j,jj} u^I_{i,jj,k}, \qquad \frac{\partial^2 u^I}{\partial \eta^2} \simeq H^I_{2,2} \equiv \sum_{jj=0}^{n} E^n_{j,jj} u^I_{i,jj,k} \tag{2.9}$$

where $D^n_{j,jj}$ and $E^n_{j,jj}$ are the first- and second-derivative pseudospectral collocation matrices using the points (2.6). In the ζ direction using all the fictitious nodes we have

$$\frac{\partial u^I}{\partial \zeta} \simeq \sum_{kk=-m}^{m} D^{2m}_{k,kk} u^I_{i,j,kk}. \tag{2.10}$$

Noting that $u^I_{i,j,k}$ is symmetric for $I = 1, 2$ and antisymmetric for $I = 3$ we have

$$\frac{\partial u^I}{\partial \zeta} \simeq G^I_3 \equiv \sum_{kk=-0}^{m}{}' \left(D^{2m}_{k,kk} \pm D^{2m}_{k,2m-kk}\right) u^I_{i,j,kk} \tag{2.11}$$

where "+" is used for $I = 1, 2$ and "−" for $I = 3$ and $\sum'$ denotes a summation whose first term is halved. Similarly,

$$\frac{\partial^2 u^I}{\partial \zeta^2} \simeq H^I_{3,3} \equiv \sum_{kk=-0}^{m}{}' \left(E^{2m}_{k,kk} \pm E^{2m}_{k,2m-kk}\right) u^I_{i,j,kk}. \tag{2.12}$$

The second-order mixed derivatives are obtained by applying the appropriate first derivatives twice and are

$$\frac{\partial^2 u^I}{\partial \xi \partial \eta} \simeq H^I_{1,2} \equiv \frac{1}{2} \sum_{jj=0}^{n} D^n_{j,jj} \left(u^I_{i+1,jj,k} - u^I_{i-1,jj,k}\right), \tag{2.13a}$$

$$\frac{\partial^2 u^I}{\partial \xi \partial \zeta} \simeq H^I_{1,3} \equiv \frac{1}{2} \sum_{kk=-0}^{m}{}' \left(D^{2m}_{k,kk} \pm D^{2m}_{k,2m-kk}\right) \left(u^I_{i+1,jj,k} - u^I_{i-1,jj,k}\right), \tag{2.13b}$$

$$\frac{\partial^2 u^I}{\partial \eta \partial \zeta} \simeq H^I_{2,3} \equiv \frac{1}{2} \sum_{jj=0}^{n} \sum_{kk=-0}^{m}{}' D^n_{j,jj} \left(D^{2m}_{k,kk} \pm D^{2m}_{k,2m-kk}\right) u^I_{i+1,jj,kk}. \tag{2.13c}$$

For the pressure we have

$$\frac{\partial p}{\partial \xi} \simeq P_1 \equiv p_{i+\frac{1}{2},j,k} - p_{i-\frac{1}{2},j,k}, \tag{2.14a}$$

$$\frac{\partial p}{\partial \eta} \simeq P_2 \equiv \frac{1}{2} \sum_{jj=0}^{n-1} F^{n-1}_{j,jj} \left(p_{i+\frac{1}{2},jj,k} + p_{i-\frac{1}{2},jj,k}\right), \tag{2.14b}$$

$$\frac{\partial p}{\partial \zeta} \simeq P_3 \equiv \frac{1}{2} \sum_{kk=0}^{m-1}{}' \left(F^{2m-2}_{k,kk} + F^{2m-2}_{k,2m-kk}\right) \left(p_{i+\frac{1}{2},j,kk} + p_{i-\frac{1}{2},j,kk}\right), \tag{2.14c}$$

where $F^{n-1}_{j,jj}$ is the first-derivative collocation matrix using the points (2.6) excluding $j = n$ since there is no pressure collocation point on the boundary. Because p is not defined at integer points in the ξ direction, it has been represented by the average of its two nearest neighbors. Substituting equations (2.7)-(2.14) into (2.5b) we obtain the difference equations for the momentum balance at each interior node as

$$\left(Ru^J - \frac{\partial^2 \xi^K}{\partial x^J x^J}\right) G^I_K + P_K \frac{\partial \xi^K}{\partial x^I} - H^I_{K,L} \frac{\partial \xi^K}{\partial x^J} \frac{\partial \xi^L}{\partial x^J} = 0 \tag{2.15}$$

where u^I and the derivatives of ξ^K are evaluated at i, j, k and the repeated suffices J and K are summed from 1 to 3.

The continuity equation (2.5a) is to be satisfied at the pressure collocation points. Hence the derivatives at $(i+\frac{1}{2}, j, k)$ are approximated by

$$\frac{\partial u^I}{\partial \xi} \simeq u^I_{i+1,j,k} - u^I_{i,j,k}, \tag{2.16a}$$

$$\frac{\partial u^I}{\partial \eta} \simeq \frac{1}{2}\sum_{jj=0}^{n} D^n_{j,jj}\left(u^I_{i+1,jj,k} + u^I_{i,jj,k}\right), \tag{2.16b}$$

$$\frac{\partial u^I}{\partial \zeta} \simeq \frac{1}{2}\sum_{kk=0}^{m}{}' \left(D^{2m}_{k,kk} + D^{2m}_{k,2m-kk}\right)\left(u^I_{i+1,j,kk} + u^I_{i,j,kk}\right). \tag{2.16c}$$

Substituting into (2.5a) gives

$$\begin{aligned}&\left(u^J_{i+1,j,k} - u^J_{i,j,k}\right)\frac{\partial \xi}{\partial x^J} + \frac{1}{2}\sum_{jj=0}^{n} D^n_{j,jj}\left(u^J_{i+1,jj,k} + u^J_{i,jj,k}\right)\frac{\partial \eta}{\partial x^J}\\ &+\frac{1}{2}\sum_{kk=0}^{m}{}' \left(D^{2m}_{k,kk} + D^{2m}_{k,2m-kk}\right)\left(u^J_{i+1,j,kk} + u^J_{i,j,kk}\right)\frac{\partial \zeta}{\partial x^J} = 0\end{aligned} \tag{2.17}$$

where the repeated superscript J is summed from 1 to 3. The discretized equations (2.15) and (2.17) are applied to the interior points of each grid. In order to complete the system of equations we now consider the boundary conditions.

3. Boundary conditions. Referring to Fig. 1 the boundary conditions are as follows.

(i) CD and FGH. On these boundaries we have the no-slip condition $u^I = 0$; that is

$$u^I_{i,j,k} = 0. \tag{3.1}$$

(ii) AH and DEF. At the ends of the tubes we will suppose that the inlet and outlet pressures are prescribed; that is $p = P_s$ and $p = P_f$ on AH and DEF, respectively. For the numerical difference scheme to be closed we need to supply additional velocity boundary conditions. Assuming that far upstream and downstream the tube, and consequently the flow, is uniform we will impose velocity boundary conditions $\partial u/\partial x = 0$ and $v = w = 0$. Numerically these give

$$u^1_{0,j,k} - u^1_{1,j,k} = 0, \qquad u^1_{l+1,j,k} - u^1_{l,j,k} = 0, \tag{3.2a}$$

$$u^2_{0,j,k} = u^3_{0,j,k} = 0, \qquad u^2_{l+1,j,k} = u^3_{l+1,j,k} = 0, \tag{3.2b}$$

$$p_{l+\frac{1}{2},j,k} - P_s = 0, \qquad p_{l+\frac{1}{2},j,k} - P_f = 0. \tag{3.2c}$$

(iii) AB and BE. These are interior boundaries and hence require that the dependent variables should be continuous and have continuous normal derivatives. For BE this is

$$\left.\begin{aligned} g_2 &= g_3\\ \frac{\partial g_2}{\partial y_2} &= -\frac{\partial g_3}{\partial y_3}\end{aligned}\right\} g = u, w, p \qquad \begin{aligned} v_2 &= v_3\\ \frac{\partial v_2}{\partial y_2} &= -\frac{\partial v_3}{\partial y_3}\end{aligned} \tag{3.3}$$

where the suffixes 1, 2, and 3 refer to grids I, II, and III in Fig. 1. The y derivatives are calculated numerically at $j = 0$ using

$$\frac{\partial g}{\partial y} = \frac{\partial \xi}{\partial y}\frac{\partial g}{\partial \xi} + \frac{\partial \eta}{\partial y}\frac{\partial g}{\partial \eta} + \frac{\partial \zeta}{\partial y}\frac{\partial g}{\partial \zeta} \tag{3.4}$$

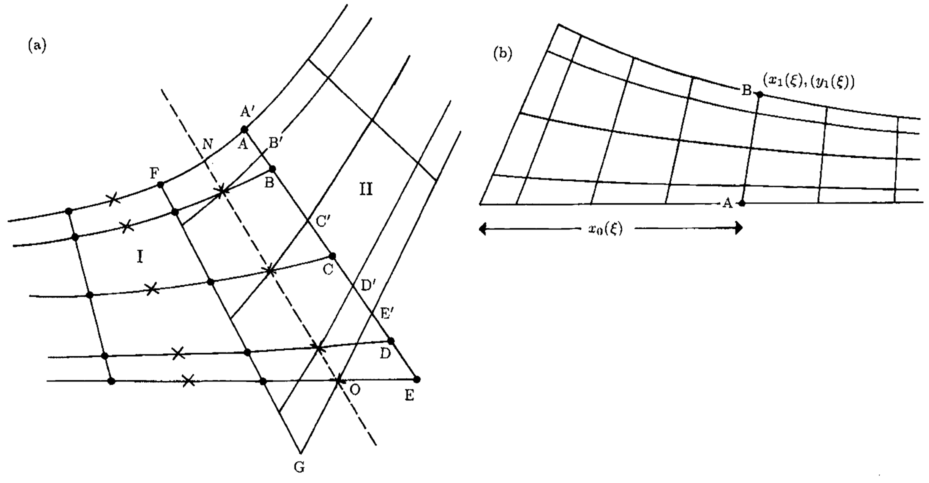


FIG. 3. (a): *The geometry of the overlap region between domains* I *and* II. (b): *The definition of* $x_0(\xi)$, $x_1(\xi)$, *and* $y_1(\xi)$ *illustrated.*

where $\partial\xi/\partial y$, $\partial\eta/\partial y$, and $\partial\zeta/\partial y$ are known algebraically and $\partial g/\partial\xi$, $\partial g/\partial\eta$, and $\partial g/\partial\zeta$ are approximated using formulae similar to (2.8), (2.9), and (2.11) except that the pressure p is centered at $i+\frac{1}{2}$ rather than i (i.e., replace i by $i+\frac{1}{2}$ in these equations).

The boundary conditions on AB are similar and, taking into account the symmetry about this line, give

$$\frac{\partial g_1}{\partial y_1}=0, \quad g=u,w,p, \qquad v_1=0. \tag{3.5}$$

(iv) BC and BG. The interior boundary BG is chosen to occur at the end pressure collocation points. In order to apply the discretized momentum balance equation near this boundary it is necessary to extend the grids by half a grid spacing so that grids I and II overlap as shown in Fig. 3(a) in which BG is now labeled ON. The values of u^I at A, B, C, D, and E (illustrated as $n=4$ in Fig. 3(a)) for grid I are obtained from the values of u^I at A′, B′, C′, D′, and E′ using interpolation. In order to preserve pseudospectral accuracy in the η direction, the interpolating polynomial is of degree m using all the nodal points $(1,j,k)$, $j=0,1,\dots,m$ for fixed k. The x and y coordinates for grids I and II do not coincide and hence the velocities on grid II calculated at positions A, B, C, D, and E need to be resolved in the directions of the velocities on grid I, using the formulae

$$\begin{aligned} u_1 &= -u_2\cos\phi + v_2\sin\phi,\\ v_1 &= u_2\sin\phi + v_2\cos\phi,\\ w_1 &= w_2. \end{aligned} \tag{3.6}$$

Similarly, boundary values along FG for grid II in Fig. 3(a) are obtained from grid I using interpolation. Finally, the boundary BG in Fig. 1 is treated in a similar way to that described for BC. The two grids involved, corresponding to grids I and II for BC, will be grid III and its reflection about BG.

4. Geometry of the bifurcation. The shape of the boundaries CD and FGH in Fig. 1, together with the parameters d_1, d_2, d_3, and ϕ, are arbitrary and, hence, user specifiable. To illustrate the numerical solution we suppose that the boundaries GH, FG, and CD for the three grids are given by

$$\sinh[b(y-d)]\sinh[b(y\cos\epsilon - x\cos\epsilon - d)] = \sinh^2(bh) \tag{4.1}$$

where $d = d_1$, $h = h_2$, and $\epsilon = \phi$ for grids I and II and $d = d_2$, $h = h_2$, and $\epsilon = \pi - 2\phi$ for grid III. This produces a curve which asymptotes exponentially quickly to the lines $y = d$ and $y\cos\epsilon - x\cos\epsilon = d$; that is, the inlet and output tubes are essentially uniform away from the bifurcation. The parameter b controls the rate at which the curve approaches the asymptote, and h controls the sharpness of the "corner" at the bifurcation.

The grid employed is specified by three functions $x_0(\xi)$, the distance from the origin along the center line, and $x_1(\xi)$ and $y_1(\xi)$ representing a point on the curve (Fig. 3(b)). Because the main changes in the flow occur near $x = 0$ we require a transformation in ξ which gives close grid spacings at small x and wider spacings at large x where the flow is essentially uniform. Hence a suitable choice of function for $x_0(\xi)$ is

$$x_0(\xi) = A\sinh[k(\xi - \tfrac{1}{2})] \tag{4.2}$$

where $\xi = 0, 1, \dots, l+1$ give the velocity collocation points and $\xi = \frac{1}{2}, \frac{3}{2}, \dots, l+\frac{1}{2}$ the pressure points. If x_s is the length of grid I, Δx_0 is the width of the spacing at $x = 0$, and l_0 is the number of grid spacings in grid I, then the parameters A and k are obtained iteratively from the equations

$$Ak = \Delta x_0, \qquad A\sinh(l_0 k) = -x_s. \tag{4.3}$$

If x_f is the minimum length of grids II and III then the number of grid spacings l in these grids is the smallest integer such that the lengths of the grids is at least x_f. The functions $x_1(\xi)$ and $y_1(\xi)$ are chosen such that AB in Fig. 3(b) meets the boundary curve at right angles, these being found numerically using Newton iteration. Hence, the transformation is

$$\begin{aligned} x &= \eta x_1(\xi) + (1-\eta)x_0(\xi), \\ y &= \eta y_1(\xi), \\ z &= d_3\zeta, \end{aligned} \tag{4.4}$$

from which the transformation functions $\partial\xi^K/\partial x^J$ and $\partial^2\xi^K/\partial x^J\partial x^J$ can be found. The algebra is formidable and consequently they were obtained using Maple. In this paper we consider two grids with (i) $\phi = 60°$ and (ii) $\phi = 90°$ having parameters

$$\begin{aligned} &x_s = 3 \quad x_f = 3 + \tfrac{1}{5}R, \quad \Delta x_0 = 0.5/l_0, \quad d_1 = d_2 = d_3 = 0.5, \quad b = 4, \\ &P_s = 15(x_s + x_f), \quad h_1 = 0.2, \quad \text{(i)} \quad h_2 = 0.2, \quad \text{or} \quad \text{(ii)} \quad h_2 = 0. \end{aligned} \tag{4.5}$$

The resulting grids for $l_0 = 20$ and $m = 8$ are shown in Fig. 4.

5. Newton's method and its implementation. Equations (2.9), (2.11), and (2.12) form a closed set for the unknowns $u^I_{i,j,k}$ and $p_{i+\frac{1}{2},j,k}$ which can be written as

$$\mathbf{f}(\mathbf{u}) = 0 \tag{5.1}$$

where $\mathbf{f}$ contains the right-hand sides of the equations and $\mathbf{u}$ the unknowns $u^I_{i,j,k}$ and $p_{i+\frac{1}{2},j,k}$. Equation (2.17) has been solved using Newton's method, which is

$$\mathbf{u}^{(s+1)} = \mathbf{u}^{(s)} + \Delta\mathbf{u}^{(s)}, \qquad s = 0, 1, \dots, \tag{5.2a}$$

$$J\Delta\mathbf{u}^{(s)} = -\mathbf{f}(\mathbf{u}^{(s)}) \tag{5.2b}$$

where J is the Jacobian $\partial\mathbf{f}/\partial\mathbf{u}$ evaluated with $\mathbf{u} = \mathbf{u}^{(s)}$.

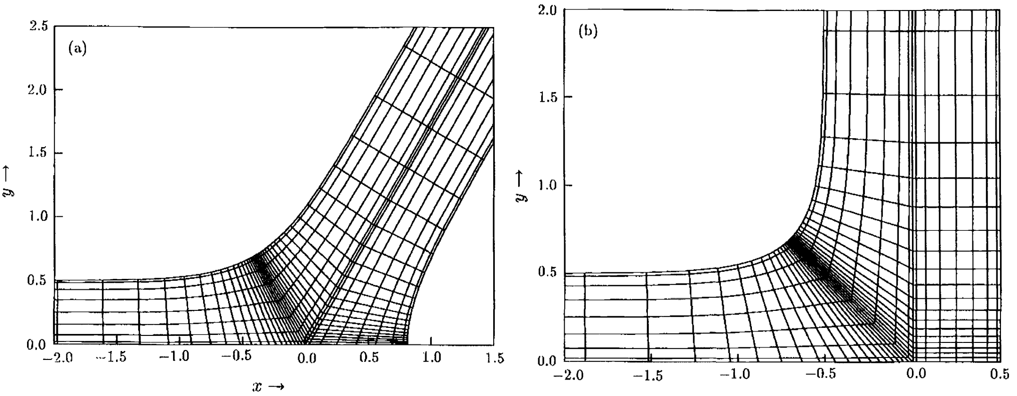


FIG. 4. *Section of the grid used in the x-y plane with* $l_0 = 20$ *and* $m = 8$ *for* (a) $\phi = 60°$ *and* (b) $\phi = 90°$.

The variables $u^l_{i,j,k}$ and $p_{i+\frac{1}{2},j,k}$ are ordered such that i is the "outermost loop" which varies the least. The order commences with the collocation points of grid I for $i = l_0 + 1, l_0 + \frac{1}{2}, l_0, \ldots, \frac{1}{2}, 0$ and the points of grids II and III are ordered together, that is, in parallel for $i = 0, \frac{1}{2}, 1, \ldots, l + \frac{1}{2}, l + 1$. This ordering causes the Jacobian J whose size $N \sim 4nm(l_0 + 2l)$ to be banded. The semibandwidth $W \sim 5nm$ for the elements resulting from the equations from grid I and $W \sim 9nm$ for grids II and III with a significant increase in width near the boundary of the two grids. Equation (5.2b) is solved using Gaussian elimination. The number of multiplications in reducing the Jacobian to triangular form $\sim W^2N$ and the storage requirement for the reduced matrix is $\sim 4WN$ bytes assuming four bytes per word. (The calculation is performed entirely in double precision, but it is sufficient to store the Jacobian on the hard disk in single precision.) For one of the cases depicted in the computer log (shown later), $l_0 = 60$, $l = 74$, $m = 14$, and $n = 7$ giving N as 83594, the maximum value for W as 1267 requiring 225.6 Mbytes for the reduced Jacobian. The reduction of the Jacobian to triangular form has been performed efficiently taking into account the "jaggedness" of the banding. The following were adopted:

(i) After each row of J is calculated, that row is immediately reduced using forward elimination and then transferred to the hard disk. This minimizes the number of disk transfers required and only the reduced Jacobian needs to be stored.

(ii) Transfers to the hard disk are done in large batches which take up less than 1 percent of the total CPU time.

(iii) The elimination process consists of three nested DO loops, with the inner loop being the most expensive. This inner loop is only performed when it makes nonzero changes, that is only for nonzero multipliers and then only up to the last nonzero element of the pivotal row.

(iv) Pivoting was not used except in the case of a zero pivot. It was found to be unnecessary (although it could be switched on if required) since pivoted and unpivoted reductions were essentially the same.

The implementation of the above resulted in a considerable reduction in the CPU time compared with the case when the "jaggedness" is not taken into account.

In order to test the validity of the code, a comparison with an analytical solution of equations (2.5) given by

$$\begin{aligned} u &= e^x \cosh y \cos \sqrt{2}z, \\ v &= e^x \sinh y \cos \sqrt{2}z, \\ w &= -\sqrt{2}e^x \cosh y \sin \sqrt{2}z, \\ p &= \tfrac{1}{2}Re^{2x}(\cos^2 \sqrt{2}z - \cosh^2 y) \end{aligned} \tag{5.3}$$

has been performed. The boundary conditions (3.1) and (3.2) are replaced by Dirichlet boundary conditions whose dependent variables are given by (5.3) and the interior discretizations (2.15) and (2.17) remain unchanged. The exact solution for the interior is then given by (5.3) and a comparison with numerical results can be made. Setting $R = 2$ gave results which demonstrated that the differenced equations (2.15) and (2.17), which form the most complex part of the program, are indeed correct.

In order to estimate the error in a particular set of results with, say, array parameters (l, m, n), three further sets of results were obtained to see the effect of altering l, m, and n in turn. To assess the effect of the choice of l on accuracy we compare the results with those obtained with half the number of grid points in the ξ direction, that is with parameters $(\frac{1}{2}l, m, n)$. Because central finite differences are used in the ξ direction, which are second order, we can estimate the error using the formula

$$\tfrac{1}{3}\left|q(l,n,m) - q(\tfrac{1}{2}l, n, m)\right| \tag{5.4}$$

where $q(l,n,m)$ and $q(\frac{1}{2}l, n, m)$ are two estimates for any variable q at a common location. For the pressure each grid location on the coarser grid coincides with a grid location on the fine grid. For the velocity, however, the points do not coincide and we use cubic interpolation to compare these values.

We can assess the effect of the choice of n and m on accuracy by obtaining results by varying these two parameters over a range of values. Numerical experimentation indicated that it is sufficient to alter these parameters by one in order to obtain a reasonable estimate of the accuracy. Hence, by obtaining results with parameters $(l, m-1, n)$ and $(l, m, n-1)$, the error due to changing each of these parameters in turn can be estimated by simple differences, that is

$$|q(l,n,m) - q(l,n-1,m)| \qquad \text{and} \qquad |q(l,n,m) - q(l,n-1,m)| \tag{5.5}$$

at a common location. Because the collocation points in the η and ζ directions do not coincide, we evaluate these differences by comparing the results at the collocation points on one grid with the interpolated solution from the other grid. Formula (5.4) estimates the error on the finer grid, but (5.5) estimates the error on the coarser and, hence, it is anticipated that the actual errors in the η and ζ directions will be significantly smaller than those given by (5.5).

To apply Newton's iteration we need a starting iterate $\mathbf{u}^{(0)}$ in (5.2). For this we choose the velocity $u^l_{i,j,k} = 0$ everywhere and $p_{i+\frac{1}{2},j,k}$ to vary linearly with x from P_s to P_f. For convergence to a specified tolerance (see later), typically four to six iterations are usually required depending on the value of R. However, for each case considered, four sets of results are obtained in order to estimate the accuracy. Hence once a set of results has been obtained an accurate starting value $\mathbf{u}^{(0)}$ is available by interpolating these results. Thus the number of iterations required in the last three runs is considerably reduced and typically two or three iterations is sufficient. Below are shown some elements of the computer log for the case $R = 20$ when $l_0 = 60$, $m = 14$, and $n = 7$.

$l_0 = 60, \quad l = 74, \quad m = 14, \quad n = 6.$
Jacobian: Size $= 71652$, semibandwidth $= 1086$, storage $= 165.8$ Mbytes.

Iteration	CPU	Error
1	4208	2.40 (+1)
2	4232	1.86 (+1)
3	4209	8.92 (−1)
4	4210	7.49 (−3)

Newton accuracy = 5.27 (−7)

$$l_0 = 60, \quad l = 74, \quad m = 14, \quad n = 7.$$

Jacobian: Size = 83594, semibandwidth = 1267, storage = 225.6 Mbytes.

Iteration	CPU	Error
1	6699	3.25 (−3)
2	6699	4.81(−6)

Newton accuracy = 1.05 (−11)

$$l_0 = 60, \quad l = 74, \quad m = 13, \quad n = 7.$$

Jacobian: Size = 77623, semibandwidth = 1176, storage = 194.2 Mbytes.

Iteration	CPU	Error
1	5354	1.40 (−3)
2	5345	4.16 (−6)

Newton accuracy = 3.69 (−11)

$$l_0 = 30, \quad l = 37, \quad m = 14, \quad n = 7.$$

Jacobian: Size = 42826, semibandwidth = 1267, storage = 114.8 Mbytes.

Iteration	CPU	Error
1	3339	1.02 (+0)
2	3338	7.24 (−3)

Newton accuracy = 3.62 (−7)

The order in which the runs have been made is chosen to reduce the CPU time as much as possible and the results (l, m, n), which take the most time per iteration, are performed second. The error recorded at each iteration is $E^{(s)} = \|\Delta\mathbf{u}^{(s)}\|_\infty$ which estimates the maximum absolute error in $\mathbf{u}^{(s)}$, that is, the previous iteration (see (5.2a)). Newton's iteration is theoretically second order and we have observed that as $E^{(s)} \to 0$ this is indeed the case to a reasonable degree of accuracy. Hence assuming that the iteration does perform as second order we can estimate the error in $\mathbf{u}^{(s+1)}$ using the formula $E_{\text{est}} = [E^{(s)}]^3/[E^{(s-1)}]^2$. The runs were terminated when either $E^{(s)} < 10^{-4}$ or $E_{\text{est}} < 10^{-6}$. In the log the item "Newton accuracy" is the value of E_{est} for the final iteration and shows that all the runs are accurate to at most 10^{-8}. Also in the log are details of the Jacobian, its size, bandwidth, and storage requirement.

It is possible to speed up the program using a quasi-Newton iteration technique in which Newton's method is applied for two or three iterations and subsequent iterations use the L and U factors from the last full iteration. This approach avoids the costly inversion of the matrix J at each iteration, and, although the iteration is now first order, the convergence is usually

TABLE 1

Central inlet and outlet velocities of the flow, daughter tube length, and end pressure of the main tube for selected values of R.

	End tube velocities					
	$\phi = 60°$		$\phi = 90°$			
R	Inlet	Outlet	Inlet	Outlet	Daughter tube length	Inlet Pressure
0	1.7035	0.8518	1.8068	0.9034	3.000	90.00
10	1.9234	0.9618	2.0157	1.0079	5.680	130.19
20	2.0448	1.0225	2.1264	1.0633	7.331	154.97
30	2.1317	1.0660	2.1932	1.0967	9.462	186.94
50	2.2145	1.1074	2.2412	1.1208	13.875	253.13
70	2.2410	1.1206	2.2463	1.1233	17.909	313.63
100	2.2461	1.1232	2.2423	1.1213	20.346	350.19

rapid. As discussed in a previous paper [11] this technique was found to save significant CPU time after the initial few full iterations had been performed. Because, for each case, the last three runs only take two or three iterations to converge, the technique could only be applied to the first run which typically requires four to six iterations. For ease of programming, quasi-Newton was not used in these calculations.

The calculations were performed on a DEC Alpha. The 0.5 factor used in the formula for Δx_0 in (4.5) was chosen because numerical experimentation indicated that this gave the most accurate results. Also the values of l, m, and n employed (see next section) are the maximum values allowed by the limits of the machine. A typical run required 16–20 hours of CPU time and up to 300 Megabytes of disk space.

6. Results and discussion. Results have been obtained for $R = 0, 10, \ldots, 100$ using $l_0 = 60$, $m = 14$, and $n = 7$ for the case $\phi = 60°$ and $l_0 = 60$, $m = 12$ and $n = 8$ for the case $\phi = 90°$. Table 1 shows the central inlet velocity of the main tube and the central outlet velocity of the daughter tubes as obtained from the results for selected values of R. Also shown are the lengths of the daughter tubes used and the pressure P_s applied at the end of the main tube. The length of the main tube x_s is set at three and the pressure at the end of the daughter tubes is zero. Tables 2 and 3 list the maximum and average errors resulting from changing the values of l, m, and n in turn as given by equations (5.4) and (5.5) for the two cases of ϕ considered. The velocity error recorded is the absolute error (since the velocity is order unity throughout the flow), but the pressure error is relative to the average drop in pressure per unit distance, i.e., $(P_s - P_f)/(x_f + x_s)$. Average errors for the velocity are at most a few times 10^{-4} and for pressure a few times 10^{-3}, with maximum errors about a magnitude higher. As expected the errors increase as the Reynolds number increases.

The results for $\phi = 60°$ with $R = 50$ are shown in Fig. 5. The diagram traces the paths of streamlines showing them from four viewpoints, namely, side elevation, end elevation, plan, and in the direction of the axis of the daughter tube. The magnitude of the velocities is indicated by the distance between arrowheads shown in the side elevation. More precisely the arc length between arrowheads is proportional to the velocity magnitude in the x-y plane, i.e., $\sqrt{u^2 + v^2}$, evaluated at the tail-end of the arrow. The arrow shown between side elevation and plan has unit velocity. As the flow passes through the bifurcation it twists in a fairly complex fashion. The nature of this twist can be shown by placing a uniform grid in the flow far upstream (Fig. 6(a)) and comparing it with the resulting distorted grid far downstream (Fig. 6(b)). The corners ABCD become mapped into A′B′C′D′. A careful examination of the diagrams shows that it is very unlikely that, at this Reynolds number, there are any

TABLE 2

Maximum and average errors in velocity and pressure for $\phi = 60°$ resulting from changes in the tube parameters l, m, and n for selected Reynolds number R.

R		Velocity			Pressure		
		l	m	n	l	m	n
0	Max	3.8 (−4)	1.5 (−5)	5.4 (−5)	1.7 (−4)	8.1 (−5)	6.7 (−5)
	Ave	4.8 (−5)	9.8 (−7)	2.8 (−6)	4.7 (−5)	5.0 (−6)	5.0 (−6)
10	Max	5.0 (−4)	1.2 (−5)	6.0 (−5)	2.8 (−4)	5.4 (−5)	5.7 (−5)
	Ave	6.6 (−5)	8.8 (−7)	2.9 (−6)	1.0 (−4)	4.4 (−6)	4.9 (−6)
20	Max	9.0 (−4)	9.7 (−6)	6.4 (−5)	7.5 (−4)	9.3 (−5)	1.3 (−4)
	Ave	9.0 (−5)	8.7 (−7)	3.4 (−6)	2.0 (−4)	5.4 (−6)	1.1 (−5)
30	Max	1.3 (−3)	2.4 (−5)	9.0 (−5)	1.7 (−3)	2.2 (−4)	5.4 (−4)
	Ave	1.2 (−4)	1.3 (−6)	5.1 (−6)	3.4 (−4)	9.9 (−6)	3.0 (−5)
50	Max	1.6 (−3)	5.2 (−4)	5.9 (−4)	4.5 (−3)	4.0 (−3)	3.2 (−3)
	Ave	1.6 (−4)	1.9 (−5)	2.6 (−5)	7.1 (−4)	1.6 (−4)	2.8 (−4)
70	Max	1.9 (−3)	2.6 (−3)	1.9 (−3)	7.9 (−3)	2.4 (−2)	1.4 (−2)
	Ave	1.9 (−4)	1.3 (−4)	9.3 (−5)	1.2 (−3)	1.1 (−3)	9.5 (−4)
100	Max	2.5 (−3)	5.7 (−3)	6.5 (−3)	1.3 (−2)	7.0 (−2)	5.9 (−2)
	Ave	2.2 (−4)	2.8 (−4)	3.3 (−4)	1.9 (−3)	2.7 (−3)	3.8 (−3)

TABLE 3

Maximum and average errors in velocity and pressure for $\phi = 90°$ resulting from changes in the tube parameters l, m, and n for selected Reynolds number R.

R		Velocity			Pressure		
		l	m	n	l	m	n
0	Max	4.6 (−4)	9.9 (−5)	3.6 (−5)	2.8 (−4)	3.7 (−4)	4.2 (−5)
	Ave	5.5 (−5)	5.8 (−6)	1.6 (−6)	6.6 (−5)	3.4 (−5)	2.2 (−6)
10	Max	6.7 (−4)	7.2 (−5)	3.9 (−5)	3.3 (−4)	2.8 (−4)	8.3 (−5)
	Ave	6.7 (−5)	4.6 (−6)	1.7 (−6)	1.2 (−4)	2.7 (−5)	4.5 (−6)
20	Max	7.7 (−4)	4.1 (−5)	4.1 (−5)	6.9 (−4)	2.9 (−4)	1.3 (−4)
	Ave	8.5 (−5)	3.9 (−6)	2.0 (−6)	2.2 (−4)	2.2 (−5)	1.0 (−5)
30	Max	1.0 (−3)	3.9 (−5)	4.7 (−5)	1.4 (−3)	3.9 (−4)	2.6 (−4)
	Ave	1.0 (−4)	4.0 (−6)	2.6 (−6)	3.0 (−4)	2.2 (−5)	1.8 (−5)
50	Max	1.2 (−3)	1.9 (−4)	9.4 (−5)	3.7 (−3)	6.8 (−4)	1.0 (−3)
	Ave	1.3 (−4)	1.2 (−5)	7.5 (−6)	4.5 (−4)	6.8 (−5)	9.1 (−5)
70	Max	1.3 (−3)	1.5 (−3)	6.1 (−4)	6.4 (−3)	6.3 (−3)	3.8 (−3)
	Ave	1.4 (−4)	6.6 (−5)	4.0 (−5)	6.8 (−4)	5.1 (−4)	4.5 (−4)
100	Max	1.6 (−3)	5.3 (−3)	2.2 (−3)	1.1 (−2)	3.0 (−2)	2.2 (−2)
	Ave	1.6 (−4)	2.8 (−4)	1.6 (−4)	1.1 (−3)	3.1 (−3)	2.3 (−3)

recirculation regions in the flow. The pressure contours on the plane of symmetry ($z = 0$) and on the plane passing through the edge $z = \frac{1}{2}$ are shown in Fig. 7. These are typically as one would expect, that is, decreasing pressure from upstream to downstream, with a local maximum at the apex ($y = z = 0$ with x on the boundary) since the flow is slow within this neighborhood.

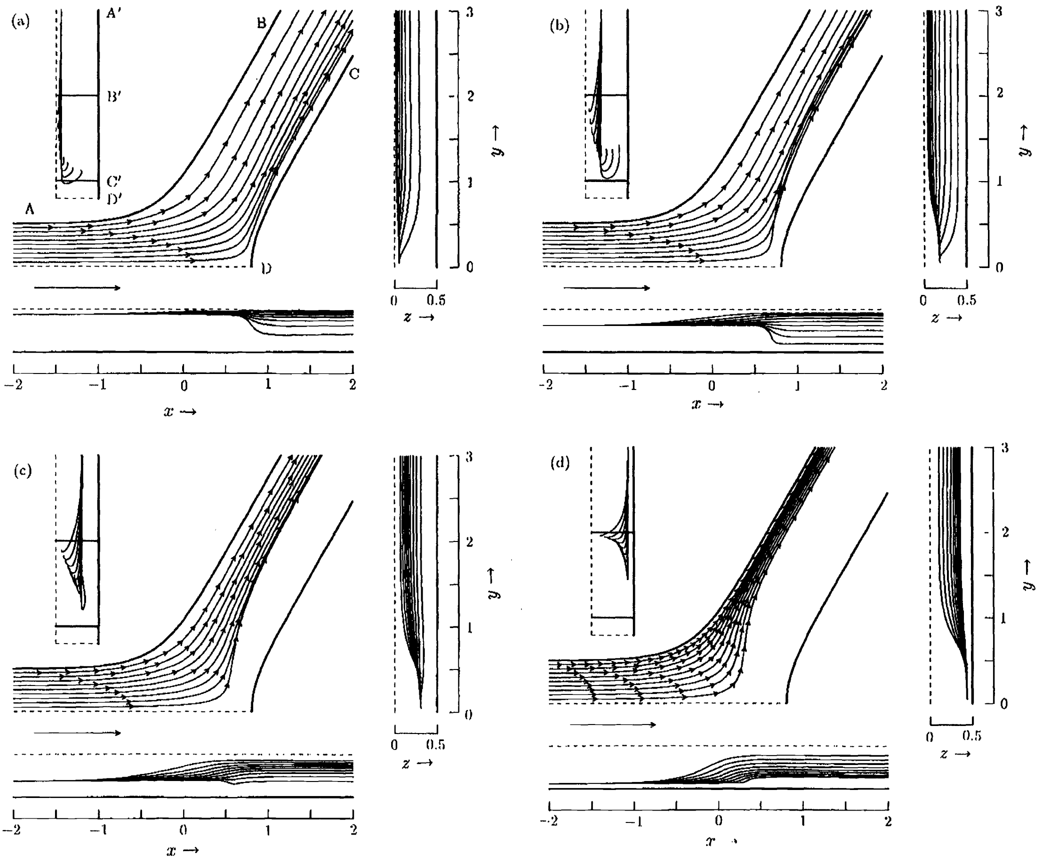


FIG. 5. *Streamlines for $\phi = 60°$ at $R = 50$. The diagram shows side elevation, end elevation, and plan. The insert at the top left shows the view along the daughter tube for which the points $A'B'C'D'$ correspond to the points ABCD in side elevation. Dashed lines are lines of symmetry. The component of the velocity magnitude in the x-y plane is shown by the distance between arrowheads in side elevation. Unit velocity is shown by the arrow between side elevation and plan. The streamlines shown commence downstream at $y = 0.1, 0.2, \ldots, 0.9$ and* (a) *at $z = 0.125$,* (b) *at $z = 0.375$,* (c) *at $z = 0.625$, and* (d) *at $z = 0.875$.*

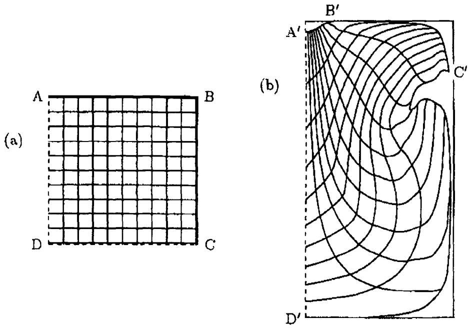


FIG. 6. *The distortion of a uniform grid far upstream:* (b) *which was originally placed perpendicularly in the flow far downstream* (a) *for $\phi = 60°$ at $R = 50$. The points ABCD are transformed into $A'B'C'D'$.*

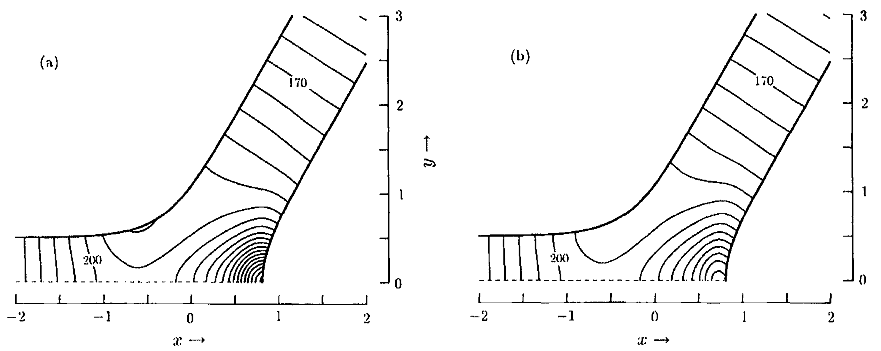


FIG. 7. *Pressure contours for* $\phi = 60°$ *at* $R = 50$ *in the plane:* (a) $z = 0$ *and* (b) $z = 0.5$. *The contour levels are equally spaced with* $\Delta p = 5$. *Two values of the contour levels are given from which the others can be determined.*

Figures 8 and 9 show the velocity streamlines and pressure contours for $\phi = 90°$ with $R = 50$. The characteristics of the flow are similar to those described for $\phi = 60°$ with the similar twisting of the flow as it passes through the bifurcation. As the Reynolds number increases, the twisting of the flow becomes much more pronounced and is illustrated in Fig. 10, which shows the streamlines of the flow for $R = 100$ and $\phi = 60°$.

7. Conclusions and future work.

(i) Numerical results have been obtained for the steady incompressible flow in a symmetric three-dimensional bifurcation having uniform thickness and specifiable boundaries in the x-y plane. This is accomplished by dividing the physical region into three domains each of which is transformed onto a rectangular parallelepiped upon which the calculations are made. The results have acceptable relative accuracy of $\sim 10^{-3}$ for Reynolds number up to 100 with daughter tubes orientated at 60° and 90° to the main tube.

(ii) The results show that there are significant flow features which do not appear in a corresponding two-dimensional flow model. In particular as the flow moves from the main tube to the daughter tubes it twists in a vortex-like fashion.

(iii) We have only considered a steady solution leaving the unsteady pulsatile case, which is a characteristic of blood flow, to a future study. However, the steady case does represent the "time averaged" situation and should be capable of predicting the areas of high/low wall shear stress where atherosclerotic lesions can occur.

(iv) Instead of using second-order differences along the tube, it would be feasible to use fourth-order central differences in order to increase the accuracy in this direction. Equations (2.8) would then encompass five nodal points rather than three with the result that the width of the Jacobian would double. This would lead to a fourfold increase in the CPU time and a doubling of the storage requirement. To be competitive with second-order differences it would be necessary to decrease the number of nodes along the tube to be at least a factor four while maintaining the same level of accuracy. It is unclear whether this is the case and would require further investigation.

(v) Although only two tubes of constant thickness have been considered, the strategy can deal with any bifurcation that is symmetric in the y and z planes. The boundaries can be specified quite generally and then the coefficients $\partial\xi^K/\partial x^J$ and $\partial^2\xi^K/\partial x^J\partial x^J$ of equation (2.5) can be found using a Maple program designed for the purpose. Such a program produces Fortran coding which is then used directly in the main fluid flow calculation.

(vi) In two-dimensional flow the CPU time requirement is proportional to N^2 for both a Newton- and a Gauss–Seidel-type method where N is the total number of nodal points. In

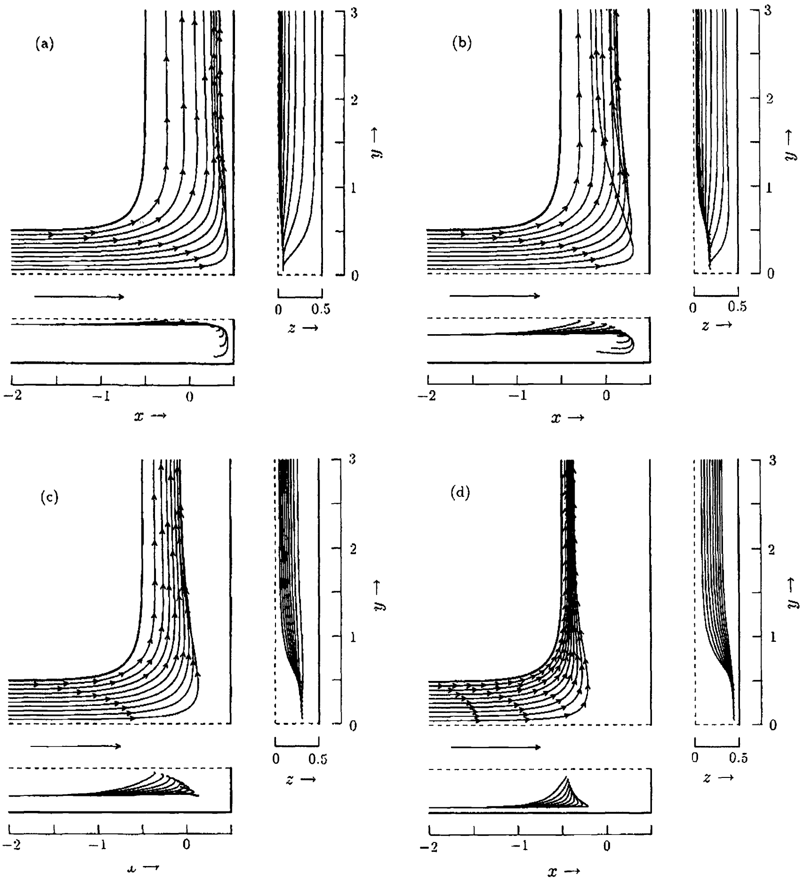


FIG. 8. *Streamlines for $\phi = 90°$ at $R = 50$. The diagram is similar to Fig. 5 except the insert is no longer necessary:* (a) *at* $z = 0.125$ *and* (b) *at* $z = 0.375$. *Streamlines for $\phi = 90°$ at $R = 50$:* (c) *at* $z = 0.625$ *and* (d) *at* $z = 0.875$.

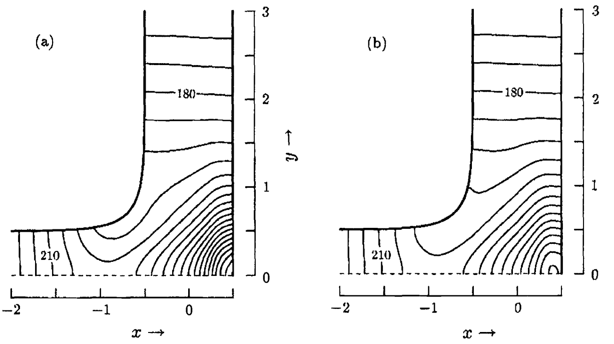


FIG. 9. *Pressure contours for $\phi = 90°$ at $R = 50$ in the plane:* (a) $z = 0$ *and* (b) $z = 0.5$. *The contour levels are equally spaced with $\Delta p = 5$.*

three dimensions the time requirement is proportional to $N^{7/3}$ and $N^{5/3}$, respectively, making Newton's method the more expensive. It is hoped that we can investigate the possibility that

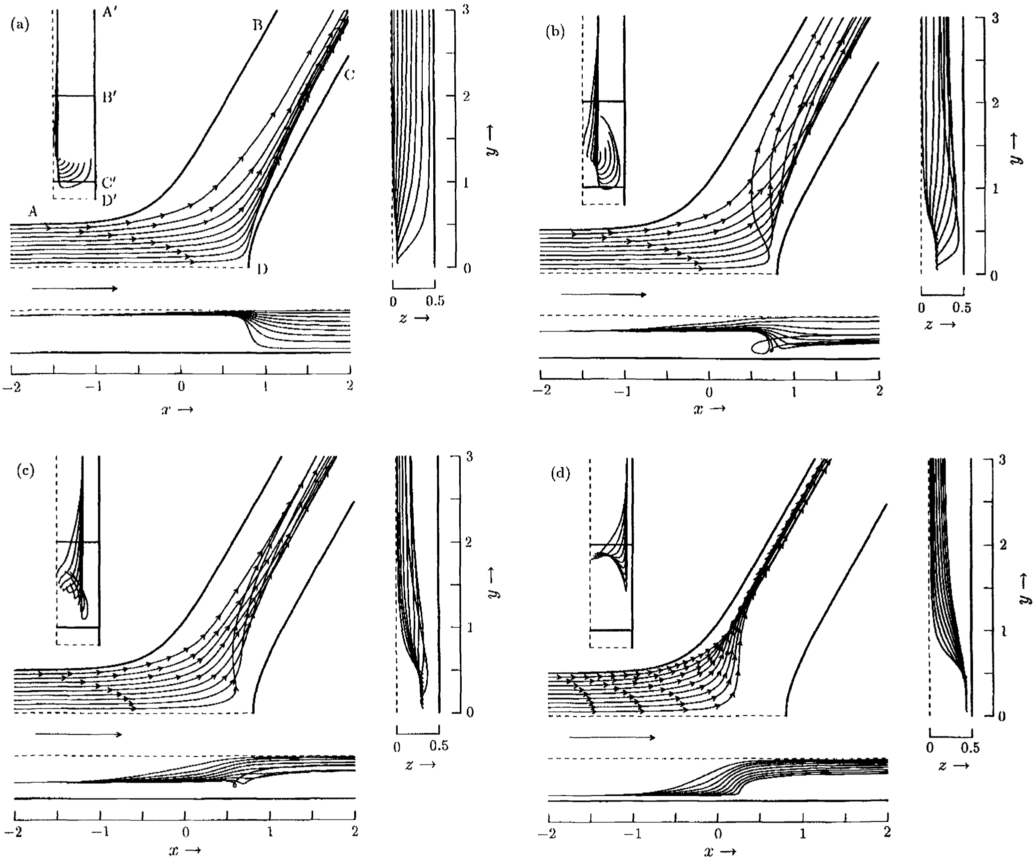


FIG. 10. *Streamlines for* $\phi = 60°$ *at* $R = 50$: (a) *at* $z = 0.125$ *and* (b) *at* $z = 0.375$. *Streamlines for* $\phi = 60°$ *at* $R = 50$: (c) *at* $z = 0.125$ *and* (d) *at* $z = 0.875$.

a combination of a Newton- and a Gauss-Seidel-type method would pick out the best of both methods in terms of speed and stability.

(vii) In the future we hope to extend the method to include cylindrical-like tubes and nonsymmetric bifurcations from which we should be able to obtain results for a realistic arterial bifurcation. Also the code needs adapting to cope with slender bifurcations (e.g., $\phi = 30°$) which gave disappointing results.

REFERENCES

[1] C. CANUTO, M. Y. HUSSAINI, A. QUARTERONI, AND T. A. ZANG, *Spectral Methods in Fluid Dynamics*, Springer-Verlag, New York, 1988.

[2] B. W. CHAR, K. O. GEDDES, H. G. GASTON, B. L. LEONG, M. B. MONOGAN, AND S. M. WATT, *Maple* V *Library Reference Manual*, Springer-Verlag, New York, 1991.

[3] B. FORNBERG, *A numerical study of steady viscous flow past a circular cylinder*, J. Fluid Mech., 98 (1980), pp. 819-855.

[4] ———, *Steady viscous flow past a circular cylinder up to Reynolds number* 600, J. Comput. Phys., 61 (1985), pp. 297-320.

[5] ———, *Steady viscous flow past a sphere at high Reynolds numbers*, J. Fluid Mech., 190 (1988), pp. 471-489.

[6] ———, *Steady incompressible flow past a row of circular cylinders*, J. Fluid Mech., 225 (1991), pp. 655-671.

[7] B. Fornberg and D. M. Sloan, *A review of pseudospectral methods for solving partial differential equations*, Acta Numerica, 1994, pp. 203-267.

[8] F. H. Harlow and J. E. Welch, *Numerical calculation of time-dependant viscous incompressible flow of fluid*, Phys. Fluids, 8 (1965), pp. 2182-2189.

[9] R. Hunt, *The numerical solution of the laminar flow in a constricted channel at moderately high Reynolds number using Newton iteration*, Internat. J. Numer. Meth. Fluids, 11 (1990), pp. 247-259.

[10] ———, *The numerical solution of the flow in a general bifurcating channel at moderately high Reynolds number using boundary-fitted co-ordinates, primitive variables and Newton iteration*, Internat. J. Numer. Meth. Fluids, 17 (1993), pp. 711-729.

[11] ———, *Three-dimensional flow in a general tube using a combination of finite and pseudospectral discretisations*, SIAM J. Sci. Comp., to appear.

[12] F. N. van de Vosse, *Numerical analysis of carotid artery flow*, Ph.D. Thesis, Eindhoven University, The Netherlands, 1987.

SIAM J. SCI. COMPUT.
Vol. 17, No. 3, pp. 579-612, May 1996

002

A STABLE PENALTY METHOD FOR THE COMPRESSIBLE NAVIER-STOKES EQUATIONS: I. OPEN BOUNDARY CONDITIONS*

J. S. HESTHAVEN†‡ AND D. GOTTLIEB†§

Abstract. The purpose of this paper is to present asymptotically stable open boundary conditions for the numerical approximation of the compressible Navier-Stokes equations in three spatial dimensions. The treatment uses the conservation form of the Navier-Stokes equations and utilizes linearization and localization at the boundaries based on these variables.

The proposed boundary conditions are applied through a penalty procedure, thus ensuring correct behavior of the scheme as the Reynolds number tends to infinity. The versatility of this method is demonstrated for the problem of a compressible flow past a circular cylinder.

Key words. open boundary conditions, stable penalty methods, Navier-Stokes equations

AMS subject classifications. 65M12, 35B35, 35Q30, 76N10

1. Introduction.

In the present paper, we discuss boundary conditions for dissipative, wave-dominated problems, exemplified by the Burgers equation and the three-dimensional, compressible Navier-Stokes equations given in conservation form. The emphasis is on deriving open boundary conditions that ensure the continuous problems are well posed and on devising asymptotically stable semi-discrete schemes for imposing these conditions. The boundary conditions and the semi-discrete schemes are valid even in the limit of vanishing viscosity.

When addressing exterior wave-dominated, dissipative problems, one is often forced to introduce an artificial boundary for computational reasons. This introduces the well-known problem of specifying appropriate boundary conditions at the artificial open boundary. For purely hyperbolic problems, it is well known that enforcing these boundary conditions through the characteristic variables leads to a stable approximation. However, for dissipative wave problems the procedure is considerably more complicated.

Naturally, we require that the boundary conditions lead to a well-posed, continuous problem. For wave problems of dissipative type, the problem must, in order to be compatible with weak boundary layers, remain well posed even in the limit where the dissipation vanishes and the problem becomes purely hyperbolic. In addition to this, we want the discrete approximation of the problem to be asymptotically stable and the boundary conditions to be easily implemented.

For general nonlinear problems the issues of well-posedness and asymptotic stability are very complicated and for most problems only very little is known. However, as discussed by Kreiss and Lorenz [1], we may, for a large class of operators, simplify the problem significantly if the solutions are smooth. It was shown that in this case it is sufficient to consider the questions of well-posedness and asymptotic stability for the locally linearized, constant coefficient version of the full problem.

The energy method is applied to the linearized, constant coefficient version of the continuous problem in order to obtain energy inequalities which bound the temporal growth of the solutions to the initial-boundary value problem. This technique allows the handling of such

*Received by the editors May 25, 1994; accepted for publication (in revised form) January 20, 1995.

†Division of Applied Mathematics, Brown University, Providence, RI 02912 (am420000@brownvm.bitnet).

‡The research of the first author was supported by Risø National Laboratory, the Danish Computing Center for Research and Education (UNI•C), and by the Danish Science Academy.

§The research of the second author was supported by AFOSR grant 93-0090, DARPA grant N00014-91-J-4016, and NSF grant DMS-9211820.

complex problems as the Navier-Stokes equations and is in general applicable to symmetrizable systems of conservation laws for which an entropy function can be defined [2].

The usual way to enforce the boundary conditions in the numerical scheme, once their proper form for the continuous problem is known, is to solve the equation in the interior of the computational domain and then enforce the boundary conditions at the boundary points. However, this approach does not take into account the fact that the equation should be obeyed arbitrarily close to the open boundary. To circumvent this problem, Funaro and Gottlieb [3, 4] and Carpenter, Gottlieb, and Abarbanel [5] developed the penalty method, which enforces the boundary conditions as well as considers the equation at the boundary. They showed asymptotic stability for the scheme applied to scalar hyperbolic equations and systems of hyperbolic equations. Don and Gottlieb [6] showed how this idea can help in applying the Legendre collocation method on the Chebyshev grids.

The proofs presented in this paper are all done for semi-discrete schemes. The relation between the stability of the semi-discrete and the fully discrete scheme was recently discussed by Kreiss and Wu [7].

The issue of well-posed boundary conditions for the compressible Navier-Stokes equations was previously considered by Gustafsson and Sundström [8], Oliger and Sundström [9], and Nordström [10, 11]. They all used the energy method to derive boundary conditions for the linearized, constant coefficient Navier-Stokes equations in the primitive variable formulation. Dutt [12] introduced an entropy function, which allowed him to derive boundary conditions for the nonlinear problem, ensuring that the solution remains bounded in an entropy norm. Halpern [13] has devised well-posed artificial boundary conditions that remain valid in the limit of vanishing viscosity. He approached the problem by viewing the linearized Navier-Stokes equations as a hyperbolic system subjected to an incomplete elliptic perturbation and obtained nonlocal as well as local open boundary conditions by considering the Fourier-Laplace transformed problem in the semi-infinite half-plane. However, for most of the previous work only few suggestions are given on how to enforce the derived boundary conditions in a natural and consistent way.

The remaining part of this paper is organized as follows. In §2 we review some well-known results on Legendre polynomials and collocation methods. Section 3 discusses the Burgers equation and boundary conditions that ensure well-posedness of the problem are derived. We continue by proposing an asymptotically stable penalty method through which the boundary conditions are enforced. This scheme ensures the correct behavior even in the limit, where the problem becomes hyperbolic, and may in general be applied to any nonlinear scalar equation. The penalty method for linear scalar hyperbolic, parabolic, and advection-diffusion equations is briefly discussed and the proposed scheme is evaluated by numerical tests. The importance of properly choosing the penalty parameter is addressed in §4, where we discuss the effect of the penalty method on the Courant-Friedrichs-Levy (CFL) condition when using explicit Runge-Kutta methods for time-stepping linear problems. We show that the results from the linear analysis carry over to the nonlinear case by performing simulations of the Burgers equation. We briefly discuss the equivalent penalty method for Chebyshev collocation methods. In §5 we derive open boundary conditions for the compressible Navier-Stokes equations given in conservation form and propose a penalty method for enforcing these boundary conditions. We derive the symmetrized form of the Navier-Stokes equation in conservation form and prove well-posedness for the continuous case and asymptotic stability of the proposed semi-discrete scheme using a Legendre collocation method. The boundary conditions and the semi-discrete approximation remain valid in the limit where the Reynolds number approaches infinity. The performance of the scheme in direct simulations of the Navier-Stokes equations is illustrated by simulating compressible flow around a circular cylinder using a Fourier-

Chebyshev collocation method. Section 6 concludes the paper and discusses the application of the scheme in finite difference/finite element simulations.

2. Legendre polynomials and collocation methods. The schemes, which we analyze in the present paper, are all based on Legendre collocation methods. This choice is dictated merely by a wish to obtain analytical results, and the methods extend trivially to other collocation methods and even to finite difference/finite element methods.

The Legendre polynomial of order N, $P_N(x)$, is defined as

$$P_N(x) = \frac{1}{2^N N!} \frac{d^N}{dx^N}(x^2-1)^N \ ,$$

where $|x| \leq 1$. In what follows, we will only consider collocation methods where the collocation points are given as the Legendre-Gauss-Lobatto points, defined as the roots of the polynomial $(1-x^2)P_N'(x)$. There is no known explicit formula for these roots.

Associated with the Gauss-Lobatto points is the quadrature formula, stating that if $f(x)$ is a polynomial of degree $2N-1$, then

$$\sum_{k=0}^{N} f(x_k)\omega_k = \int_{-1}^{1} f(\xi)\,d\xi \ , \tag{1}$$

where x_k are the Legendre-Gauss-Lobatto collocation points, and the Gauss-Lobatto weights, ω_k, are given as

$$\begin{aligned} \omega_k &= -\frac{2}{N+1}\frac{1}{P_N(x_k)\,P_{N-1}'(x_k)} \ , \quad 1 \leq k \leq N-1 \ , \\ \omega_0 = \omega_N &= \frac{2}{N(N+1)} \ . \end{aligned} \tag{2}$$

For further details on the properties of the Legendre polynomials, we refer to [14].

In a Legendre collocation method, the function, $f(x)$, is approximated by a grid function, $f_k = f(x_k)$, where the grid points are the Gauss-Lobatto collocation points. Thus, we construct a global Legendre interpolant, I_N, to obtain an approximation to the function as

$$(I_N f)(x) = \sum_{k=0}^{N} f_k\, h_k(x) \ ,$$

where the interpolating Legendre-Lagrange polynomials are given as

$$h_k(x) = -\frac{(1-x^2)P_N'(x)}{N(N+1)\,(x-x_k)P_N(x_k)} \ .$$

We note that by construction,

$$(I_N f)(x_k) = f_k \ .$$

To seek equations for an approximate solution, $(I_N f)(x)$, to a partial differential equation, we need to obtain values for the spatial derivatives at the collocation points. This is done by approximating the differential operator by a matrix operator, with the matrix entries given as

$$\mathcal{D}_{kl} = h_l'(x_k) \ .$$

For the explicit expression of the entries, we refer to [15, 16].

3. The Burgers equation. In this section, we consider the Burgers equation

$$\frac{\partial U}{\partial t} + U\frac{\partial U}{\partial x} = \varepsilon\frac{\partial^2 U}{\partial x^2} \quad |x| \leq 1 \quad t > 0 \ , \tag{3}$$

where $\varepsilon \geq 0$. The initial condition is given as

$$U(x, 0) = f(x) \ ,$$

with boundary conditions of the form

$$\alpha U(-1, t) - \beta\varepsilon \left.\frac{\partial U}{\partial x}\right|_{x=-1} = 0 \ , \tag{4}$$

$$\gamma U(1, t) + \delta\varepsilon \left.\frac{\partial U}{\partial x}\right|_{x=1} = 0 \ . \tag{5}$$

When addressing the issue of well-posedness it is sufficient to consider the linearized, constant coefficient version of the Burgers equation

$$\frac{\partial U}{\partial t} + \lambda\frac{\partial U}{\partial x} = \varepsilon\frac{\partial^2 U}{\partial x^2} \quad |x| \leq 1 \ t > 0 \ . \tag{6}$$

Here $\lambda = U_0$ is the uniform solution around which we have linearized. Equation (6) is also known as the linear advection-diffusion equation.

The four real constants, α, β, γ, and δ, in the boundary conditions, (4) and (5), may not be chosen arbitrarily, since the resulting problem should be well posed. Bounds yielding a sufficient condition for well-posedness are given in the following lemma.

LEMMA 3.1. *Equation* (6), *with boundary conditions given by* (4) *and* (5), *is well posed if one of the following conditions holds:*

(i) $\beta = 0$, $\delta = 0$.

(ii) $\beta \neq 0$, $\delta = 0$ *and* $(\varepsilon - \lambda) + 2\alpha/\beta \geq 0$.

(iii) $\beta = 0$, $\delta \neq 0$ *and* $(\varepsilon + \lambda) + 2\gamma/\delta \geq 0$.

(iv) $\beta \neq 0$, $\delta \neq 0$ *and* $2(\varepsilon - \lambda)\gamma/\delta + 2(\varepsilon + \lambda)\alpha/\beta + 4(\alpha\gamma)/(\beta\delta) \geq \lambda^2$.

Proof. Construct the energy integral as

$$\frac{1}{2}\frac{d}{dt}\|U\|^2 = -\lambda\,(U, U_x) + \varepsilon\,(U, U_{xx}) = \frac{1}{2}\left[-\lambda U^2 + 2\varepsilon U\,U_x\right]_{-1}^{1} - \varepsilon\|U_x\|^2 \ .$$

Here we have introduced

$$(U, V) = \int_{-1}^{1} U\,V\,dx \ , \quad (U, U) = \|U\|^2 \ .$$

Following an analysis similar to that in [17], we apply the following

$$-\varepsilon\|U_x\|^2 \leq -\frac{\varepsilon}{2}[U(1) - U(-1)]^2 \ .$$

Following this, the condition for well-posedness becomes

$$\frac{1}{2}\frac{d}{dt}\|U\|^2 \leq \frac{1}{2}\left[-\lambda U^2 + 2\varepsilon U\,U_x\right]_{-1}^{1} - \frac{\varepsilon}{2}[U(1) - U(-1)]^2 \leq 0 \ .$$

Condition (i) implies that $U(-1) = U(1) = 0$ such that

$$\frac{1}{2}\frac{d}{dt}\|U\|^2 \leq 0 \ .$$

For condition (ii) we obtain $U(1) = 0$ and thus

$$\frac{1}{2}\frac{d}{dt}\|U\|^2 \leq -\frac{1}{2}\left(\varepsilon - \lambda + 2\frac{\alpha}{\beta}\right)U^2(-1) \leq 0 \ ,$$

yielding the condition

$$\varepsilon - \lambda + 2\frac{\alpha}{\beta} \geq 0 \ .$$

Likewise, for condition (iii) we obtain

$$\frac{1}{2}\frac{d}{dt}\|U\|^2 \leq -\frac{1}{2}\left(\varepsilon + \lambda + 2\frac{\gamma}{\delta}\right)U^2(1) \leq 0 \ ,$$

showing that this choice yields well-posedness. For condition (iv) we obtain the constraint

$$\frac{1}{2}\frac{d}{dt}\|U\|^2 \leq -\frac{1}{2}\left(\varepsilon - \lambda + 2\frac{\alpha}{\beta}\right)U^2(-1) + \varepsilon U(-1)U(1) - \frac{1}{2}\left(\varepsilon + \lambda + 2\frac{\gamma}{\delta}\right)U^2(1) \leq 0.$$

This is obeyed if

$$\varepsilon^2 - \left(\varepsilon - \lambda + 2\frac{\alpha}{\beta}\right)\left(\varepsilon + \lambda + 2\frac{\gamma}{\delta}\right) \leq 0 \ ,$$

implying

$$2(\varepsilon - \lambda)\gamma/\delta + 2(\varepsilon + \lambda)\alpha/\beta + 4(\alpha\gamma)/(\beta\delta) \geq \lambda^2 \ . \qquad \square$$

3.1. The semi-discrete scheme. Equation (3) will be solved using a Legendre collocation method where the collocation points are the Legendre-Gauss-Lobatto points. This involves finding an Nth degree polynomial, $u(x, t)$, satisfying

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \varepsilon\frac{\partial^2 u}{\partial x^2} \quad \text{at } x = x_k \ , \quad k \in [1 \ldots N-1] \ , \tag{7}$$

in the interior. The boundary points are given by boundary conditions of Robin type

$$\alpha u(x_0, t) - \beta\varepsilon\left.\frac{\partial u}{\partial x}\right|_{x_0} = g_1(t) \ ,$$

$$\gamma u(x_N, t) + \delta\varepsilon\left.\frac{\partial u}{\partial x}\right|_{x_N} = g_2(t) \ ,$$

where $g_1(t)$ and $g_2(t)$ are prescribed boundary conditions. The traditional method of imposing the boundary conditions is to solve (7) in the interior and enforce the boundary conditions at the boundary points only. However, this approach does not take into account the fact that the equation must be obeyed arbitrarily close to the boundary. In addition to this, it has proven difficult to implement Robin boundary conditions consistently when using spectral approximations of nonlinear problems. To overcome these problems, we follow the line of thought initiated by Funaro and Gottlieb [3, 4] and propose a penalty method for approximating the Burgers equation at the Legendre-Gauss-Lobatto collocation points, $x = x_k$, $k \in [0, \ldots, N]$, as

$$\begin{aligned}\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \ & \varepsilon\frac{\partial^2 u}{\partial x^2} \\ & - \tau_1 Q^-(x)\left[\alpha u(x_0, t) - \beta\varepsilon\left.\frac{\partial u}{\partial x}\right|_{x_0} - g_1(t)\right] \\ & - \tau_2 Q^+(x)\left[\gamma u(x_N, t) + \delta\varepsilon\left.\frac{\partial u}{\partial x}\right|_{x_N} - g_2(t)\right] \ ,\end{aligned} \tag{8}$$

where

$$(9) \qquad Q^-(x) = \frac{(1-x)P_N'(x)}{2P_N'(-1)} \ , \quad Q^+(x) = \frac{(1+x)P_N'(x)}{2P_N'(1)} \ .$$

These two functions have the property of being zero at all the Legendre-Gauss-Lobatto collocation points except at the two endpoints of the domain. Although Q^- and Q^+ here are defined as delta-functions at the boundary, we may also choose other definitions. As shown by Don and Gottlieb [6], this approach may also be applied for implementing Legendre methods at Chebyshev grids.

We note here that the penalty method as given by (8) combines the boundary conditions and the governing equation into one equation. When using the penalty method, the boundary conditions are only enforced weakly at the boundary. However, the method remains spectrally accurate, as we will soon illustrate. One may also observe that the scheme is equivalent to the traditional way of imposing boundary conditions as τ_1, τ_2 approach infinity.

The parameters, τ_1 and τ_2, are then to be determined such that the semi-discrete approximation to the initial-boundary value problem is asymptotically stable.

In order to obtain the energy inequality, we consider only homogeneous boundary conditions. As discussed in [1], this is not a restriction, since we may always introduce a variable transform such the boundary conditions become homogeneous. In the following lemma we state the bounds on τ_1 and τ_2 which ensure that the linearized, constant coefficient version of (8) is asymptotically stable.

LEMMA 3.2. *Assume $u(x,t)$ exists and let $\tau^-_{a,b}$ and $\tau^+_{a,b}$ be defined as*

$$\tau^-_{a,b} = \frac{1}{\omega\varepsilon b}[\varepsilon + 2\kappa - 2\sqrt{\kappa^2 + \varepsilon\kappa - 1/2\varepsilon\omega|\lambda|}] \ ,$$

$$\tau^+_{a,b} = \frac{1}{\omega\varepsilon b}[\varepsilon + 2\kappa + 2\sqrt{\kappa^2 + \varepsilon\kappa - 1/2\varepsilon\omega|\lambda|}] \ ,$$

where $\kappa = \omega a/b$ and

$$\omega = \frac{2}{N(N+1)}$$

is the Legendre weight at the endpoints.

If

$$\tau^-_{\alpha,\beta} \le \tau_1 \le \tau^+_{\alpha,\beta} \ ,$$

$$\tau^-_{\gamma,\delta} \le \tau_2 \le \tau^+_{\gamma,\delta} \ ,$$

then the linearized, constant coefficient version of (8) *is asymptotically stable and the solution is bounded as*

$$\frac{1}{2}\frac{d}{dt}\|u\|_N^2 \le -\varepsilon \sum_{k=1}^{N-1} \left(\frac{\partial u}{\partial x}(x_k)\right)^2 \omega_k \ .$$

Proof. We start by defining the discrete, weighted inner product as

$$(u,\ v)_N = \sum_{k=0}^{N} u(x_k)\, v(x_k)\, \omega_k \ \ , (u,u)_N = \|u\|_N^2$$

and note that since we are using a Legendre collocation method, we have, through (1), the identity

$$(u, v_x)_N = (U, V_x) \ .$$

This makes it straightforward to apply partial differentiation. Following the results stated previously, it is sufficient to obtain the energy estimate for the linearized, constant coefficient version of (8) with homogeneous boundary conditions:

$$\begin{aligned}\frac{1}{2}\frac{d}{dt}\|u\|_N^2 =& -\frac{\lambda}{2}[u^2]_{-1}^{1} + \varepsilon[u\,u_x]_{-1}^{1} - \varepsilon\|u_x\|_N^2 \\ & -\tau_1\omega u(-1)[\alpha u(-1) - \beta\varepsilon u_x(-1)] - \tau_2\omega u(1)[\gamma u(1) + \delta\varepsilon u_x(1)] \ .\end{aligned}$$

Here the subscripts designate differentiation and ω is the Legendre weight at the endpoints (2). Using the quadrature rule allows for rewriting as

$$\|u_x\|_N^2 = u_x^2(-1)\omega + u_x^2(1)\omega + \sum_{k=1}^{N-1} u_x^2(x_k)\omega_k \ .$$

Contrary to the approach followed by Funaro and Gottlieb [3, 4], we recast the problem of stability into an algebraic eigenvalue problem. For the present problem, this may seem an additional complication. However, we find that for more complicated problems this approach greatly simplifies the proofs.

Isolating the terms contributing to stability at each boundary, we obtain two conditions for asymptotic stability:

$$\mathbf{u}_-^T\mathcal{H}^-\mathbf{u}_- \le 0 \ , \quad \mathbf{u}_+^T\mathcal{H}^+\mathbf{u}_+ \le 0 \ ,$$

where $\mathbf{u}_- = [u(-1), u_x(-1)]^T$, $\mathbf{u}_+ = [u(1), u_x(1)]^T$, and

$$\mathcal{H}^- = \frac{1}{2}\begin{bmatrix} \lambda - 2\alpha\omega\tau_1 & -\varepsilon(1-\beta\omega\tau_1) \\ -\varepsilon(1-\beta\omega\tau_1) & -2\varepsilon\omega \end{bmatrix} ,$$

$$\mathcal{H}^+ = \frac{1}{2}\begin{bmatrix} -\lambda - 2\gamma\omega\tau_2 & \varepsilon(1-\delta\omega\tau_2) \\ \varepsilon(1-\delta\omega\tau_2) & -2\varepsilon\omega \end{bmatrix} .$$

Since both matrices are symmetric, the problem is reduced to ensuring that $\mathcal{H}^-$ and $\mathcal{H}^+$ are negative, semi-definite. The eigenvalues of the two matrices are found to be

$$\rho_{1,2}(\mathcal{H}^-) = \frac{1}{8}\left(-\zeta^- \pm \sqrt{(\zeta^-)^2 + 16\varepsilon(\beta^2\omega^2\varepsilon\tau_1^2 - 2\omega(\beta\varepsilon + 2\alpha\omega)\tau_1 + 2\omega\lambda + \varepsilon)}\right) ,$$

$$\rho_{1,2}(\mathcal{H}^+) = \frac{1}{8}\left(-\zeta^+ \pm \sqrt{(\zeta^+)^2 + 16\varepsilon(\delta^2\omega^2\varepsilon\tau_2^2 - 2\omega(\delta\varepsilon + 2\gamma\omega)\tau_2 - 2\omega\lambda + \varepsilon)}\right) ,$$

where $\zeta^- = -2\lambda + 4\omega\varepsilon + 4\alpha\omega\tau_1$ and $\zeta^+ = 2\lambda + 4\omega\varepsilon + 4\gamma\omega\tau_2$. It is evident that negative semi-definiteness is ensured, provided $\zeta^- > 0$ and $\zeta^+ > 0$, if

$$\begin{aligned}\beta^2\omega^2\varepsilon\tau_1^2 - 2\omega(\beta\varepsilon + 2\alpha\omega)\tau_1 + 2\omega\lambda + \varepsilon \le 0 \ , \\ \delta^2\omega^2\varepsilon\tau_2^2 - 2\omega(\delta\varepsilon + 2\gamma\omega)\tau_2 - 2\omega\lambda + \varepsilon \le 0 \ .\end{aligned}$$

The roots of the two polynomials are

$$\tau_1^{\pm} = \frac{1}{\omega\varepsilon\beta}\left(\varepsilon + 2\kappa_- \pm 2\sqrt{\kappa_-^2 + \varepsilon\kappa_- - 1/2\varepsilon\omega\lambda}\right) ,$$

$$\tau_2^{\pm} = \frac{1}{\omega\varepsilon\delta}\left(\varepsilon + 2\kappa_+ \pm 2\sqrt{\kappa_+^2 + \varepsilon\kappa_+ + 1/2\varepsilon\omega\lambda}\right) ,$$

where $\kappa_- = \omega\alpha/\beta$ and $\kappa_+ = \omega\gamma/\delta$. We introduce

$$\tau^-_{a,b} = \frac{1}{\omega\varepsilon b}[\varepsilon + 2\kappa - 2\sqrt{\kappa^2 + \varepsilon\kappa - 1/2\varepsilon\omega|\lambda|}] \ ,$$
$$\tau^+_{a,b} = \frac{1}{\omega\varepsilon b}[\varepsilon + 2\kappa + 2\sqrt{\kappa^2 + \varepsilon\kappa - 1/2\varepsilon\omega|\lambda|}] \ ,$$

where $\kappa = \omega a/b$. Since

$$\tau^-_{a,b} \geq \frac{|\lambda|}{2a\omega} + \frac{1}{4}\varepsilon\frac{1}{\omega^2} \ ,$$

for $\varepsilon \ll 1$, this ensures $\zeta^- > 0$ and $\zeta^+ > 0$.

Hence, stability is ensured for

$$\tau^-_{\alpha,\beta} \leq \tau_1 \leq \tau^+_{\alpha,\beta} \ ,$$
$$\tau^-_{\gamma,\delta} \leq \tau_2 \leq \tau^+_{\gamma,\delta} \ ,$$

with the solution being bounded as

$$\frac{1}{2}\frac{d}{dt}\|u\|^2_N \leq -\varepsilon\sum_{k=1}^{N-1} u^2_x(x_k)\omega_k \ . \qquad \square$$

3.1.1. Remarks on the penalty method for linear equations. The results stated in Lemma 3.2 allows us to derive the appropriate penalty parameter for a large class of linear equations. We consider the general linear advection-diffusion equation, (6), with the Robin boundary conditions given in (4) and (5). Solving this problem by a penalty method, equivalent to that given by (8), requires bounds on the penalty parameters in order to ensure stability of the scheme.

In what follows we will give these bounds for reference and will return to the numerical validation of these results in §4. Some of these results may be found in [3, 4, 6], but are given here in a more general framework. Remember that $\omega^{-1} \sim \mathcal{O}(N^2)$.

Hyperbolic equations. ($\varepsilon = 0.$)

1. $\lambda > 0$. Well-posedness is ensured by choosing $\alpha > 0$ and $\beta = \gamma = \delta = 0$. Thus, for this case we will only need bounds on τ_1:

$$\tau^-_{\alpha,0} = \frac{\lambda}{2\omega\alpha} \ , \quad \tau^+_{\alpha,0} = \infty \ .$$

The scheme for the hyperbolic case is stable for

$$\infty \geq \tau_1 \geq \frac{\lambda}{2\omega\alpha} \ .$$

2. $\lambda < 0$. Well-posedness is ensured by choosing $\gamma > 0$ and $\alpha = \beta = \delta = 0$. Thus, for this case we will only need bounds on τ_2:

$$\tau^-_{\gamma,0} = \frac{|\lambda|}{2\omega\gamma} \ , \quad \tau^+_{\gamma,0} = \infty \ .$$

The scheme for the hyperbolic case is stable for

$$\infty \geq \tau_2 \geq \frac{|\lambda|}{2\omega\gamma} \ .$$

Parabolic equations. ($\lambda = 0,\ \varepsilon > 0$.) Necessary and sufficient conditions for well-posedness may be obtained by choosing the four parameters, $\alpha,\ \beta,\ \gamma$, and δ, properly as stated in Lemma 3.1 [17]. We only state the results for the bounds of τ_1, since the results for τ_2 are equivalent.

1. Dirichlet boundary condition ($\alpha > 0,\ \beta = 0$).

$$\tau^-_{\alpha,0} = \frac{\varepsilon}{4\alpha}\frac{1}{\omega^2}\ ,\quad \tau^+_{\alpha,0} = \infty\ .$$

Stability is ensured for

$$\infty \geq \tau_1 \geq \frac{\varepsilon}{4\alpha}\frac{1}{\omega^2}\ .$$

2. Neumann boundary condition ($\alpha = 0,\ \beta > 0$).

$$\tau^-_{0,\beta} = \frac{1}{\beta\omega}\ ,\quad \tau^+_{0,\beta} = \frac{1}{\beta\omega}\ .$$

Stability is ensured for

$$\tau_1 = \frac{1}{\beta\omega}\ .$$

3. Robin boundary condition ($\alpha > 0,\ \beta > 0$).

$$\tau^-_{\alpha,\beta} = \frac{1}{\omega\varepsilon\beta}[\varepsilon + 2\kappa - 2\sqrt{\kappa^2 + \varepsilon\kappa}]\ ,$$
$$\tau^+_{\alpha,\beta} = \frac{1}{\omega\varepsilon\beta}[\varepsilon + 2\kappa + 2\sqrt{\kappa^2 + \varepsilon\kappa}]\ ,$$

where $\kappa = \omega\alpha/\beta$. Stability is ensured for

$$\tau^+_{\alpha,\beta} \geq \tau_1 \geq \tau^-_{\alpha,\beta}\ .$$

Advection-diffusion equations. ($\lambda \neq 0,\ \varepsilon \geq 0$). Again we must ensure well-posedness by proper choice of the four parameters as given by Lemma 3.1. We only state the results for the bounds of τ_1 since the results for τ_2 are equivalent.

1. Dirichlet boundary condition ($\alpha > 0,\ \beta = 0$).

$$\tau^-_{\alpha,0} = \frac{|\lambda|}{2\alpha}\frac{1}{\omega} + \frac{\varepsilon}{4\alpha}\frac{1}{\omega^2}\ ,\quad \tau^+_{\alpha,0} = \infty\ .$$

Stability is ensured for

$$\infty \geq \tau_1 \geq \frac{|\lambda|}{2\alpha}\frac{1}{\omega} + \frac{\varepsilon}{4\alpha}\frac{1}{\omega^2}\ .$$

2. Neumann boundary condition ($\alpha = 0,\ \beta > 0$).

$$\tau^-_{0,\beta} = \frac{1}{\beta\omega} - \sqrt{\frac{2|\lambda|\omega}{\epsilon\beta^2}}\ ,\quad \tau^+_{0,\beta} = \frac{1}{\beta\omega} + \sqrt{\frac{2|\lambda|\omega}{\epsilon\beta^2}}\ .$$

Stability is ensured for

$$\frac{1}{\beta\omega} + \sqrt{\frac{2|\lambda|\omega}{\epsilon\beta^2}} \geq \tau_1 \geq \frac{1}{\beta\omega} - \sqrt{\frac{2|\lambda|\omega}{\epsilon\beta^2}}\ .$$

3. Robin boundary conditions, $\alpha > 0,\ \beta > 0$. Results are given in Lemma 3.2.

3.2. Numerical tests. Because we aim to solve the full nonlinear Burgers equation, and not the linearized, constant coefficient version, we need to validate the results obtained from the linear analysis. We have solved the Burgers equation using the scheme given by (8) and employ a standard Legendre collocation method as described in §2 [15, 18].

The Burgers equation, (3), has a rightward traveling wave solution (see, e.g., [1]) of the form

$$U(x,t) = -a \tanh\left(a\frac{x-ct}{2\varepsilon}\right) + c \quad , x \in [-\infty, \infty] \quad , t \geq 0 \quad , \tag{10}$$

where the free-stream values

$$\lim_{x\to-\infty} U(x,t) = b_{-\infty} \quad , \quad \lim_{x\to\infty} U(x,t) = b_{\infty} \quad ,$$

are associated with the wave-speed, c, and the constant, $a \geq 0$, as

$$c = \frac{b_{-\infty} + b_{\infty}}{2} \quad , \quad a = \frac{b_{-\infty} - b_{\infty}}{2} \quad .$$

Following the results in Lemma 3.1 (condition (iv): $\alpha = \lambda,\ \beta = 1,\ \gamma = 0,\ \delta = 1$), we expect the nonlinear problem to be well posed for boundary conditions of the type

$$\lambda U(-1,t) - \varepsilon\frac{\partial U(-1,t)}{\partial x} = g_1(t) \quad , \quad \varepsilon\frac{\partial U(1,t)}{\partial x} = g_2(t) \quad ,$$

where $\lambda \geq 0$ is the value around which we have linearized locally. In the present study we have used the free-stream value at the inflow, i.e., $\lambda = b_{-\infty}$. Since we know an exact solution, the boundary conditions may be given exactly at all times using (10). As the initial condition we use

$$U(x,0) = -a \tanh\left(a\frac{x}{2\varepsilon}\right) + c \quad .$$

The solution is time-stepped using a classical fourth-order Runge-Kutta method, where the boundary conditions are imposed at the intermediate time levels.

Using the values of the penalty parameters given in Lemma 3.2 results in a stable scheme. However, the CFL number, relating the maximum allowable time-step to the spatial resolution as

$$\Delta t_{\max} \leq \text{CFL} \times \min_k \left[\frac{|u(x_k)|}{\Delta_k x} + \frac{\varepsilon}{(\Delta_k x)^2}\right]^{-1} \quad ,$$

will have to be very small in order to ensure stability. Here $|u(x_k)|$ signifies the local absolute value of u and $\Delta_k x$ represents the local grid spacing. Thus, with the theoretical value of the penalty parameter, the proposed method compares unfavorably with the traditional method because of severe time-step restrictions. Fortunately, the limits of the penalty parameters, in between which asymptotic stability is ensured, are obtained as a result of a conservative energy estimate and, hence, are not strict bounds.

We have used the values of the penalty parameter (see Lemma 3.2) as

$$\tau_1 = \frac{\tau_{\lambda,1}^-}{4} \quad , \quad \tau_2 = \frac{\tau_{0,1}^-}{4} \quad .$$

These values are found to lead to a stable scheme, provided the cell Reynolds number $\text{Re}_N = \lambda/(\varepsilon N^2) \ll 1$. The constraint simply states that increasing the Reynolds number

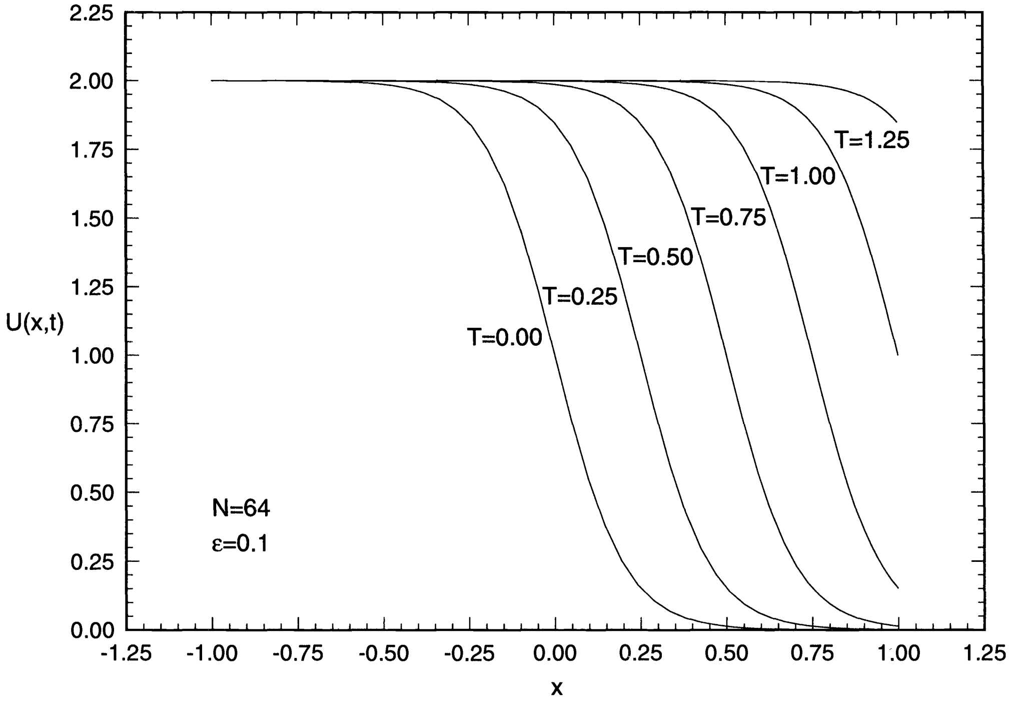


FIG. 1. *Traveling wave solution of the Burgers equation.*

TABLE 1

Error in the spectral simulation of the Burgers equation using the penalty method. The maximum error (L_∞) occurs at the boundary.

N	L_2	L_∞
16	3.41E-03	3.26E-02
32	2.43E-05	3.50E-04
64	1.09E-09	2.21E-08
128	4.98E-12	7.62E-11

requires increased spatial resolution, which is a natural restriction. For advection-dominated problems, stability is obtained by increasing the penalty parameters toward the values stated in Lemma 3.2.

With these values of the penalty parameters, we have been able to perform the simulations with a CFL number of 4, which is equivalent to what is usually allowed when using a traditional method. Thus, by fine-tuning the penalty parameters we were able to avoid any effect of the penalty method on the CFL condition. The following section contains a study of the effect of the penalty method on the CFL condition and guidelines for fine-tuning the penalty parameter for practical applications.

In Fig. 1 we show the temporal evolution of the traveling wave solution when using the scheme given by (8). The simulation is done with $N = 64$ and $\varepsilon = 0.1$. We observe no spurious reflections from the open boundary and the kink is seen to travel undisturbed out of the domain. Table 1 shows the error at $T = 1.00$, where the kink has propagated halfway through the boundary. It is evident that the proposed scheme maintains the spectral accuracy. The time-step is small enough to neglect time-stepping errors.

4. CFL restrictions for the penalty method. As discussed briefly in the previous section, choosing too large a penalty parameter results in severe CFL restrictions. For this reason, it is vital to understand how the penalty method alters the eigenvalue spectrum of the operators and consequently changes the CFL restriction.

In the present section we will study these effects for the linear advection and diffusion operators for Legendre collocation methods. For completeness, we will also give the results for Chebyshev collocation methods, which are widely used for solving nonlinear partial differential equations. The analysis will include both third- and fourth-order Runge-Kutta methods, which are often employed when addressing problems of the type considered here. At the end of the section we will compare the results from our linear analysis with simulations of the nonlinear Burgers equation.

Consider now the semi-discrete linear, constant coefficient problem

$$\begin{array}{llll} (\mathbf{q})_t = \mathcal{L}_N\mathbf{q} & x_k \in \Omega & , & t \geq 0 \ , \\ \mathbf{q} = 0 & x_k \in \Omega & , & t = 0 \ , \\ \mathcal{B}_N\mathbf{q} = 0 & x_k \in \Gamma & , & t \geq 0 \ , \end{array} \tag{11}$$

where $\mathbf{q} = (\mathbf{q}(x_0), \ldots, \mathbf{q}(x_N))^T$, $\mathcal{L}_N$ is the discrete approximation of the continuous operator for the interior, $\mathcal{L}$, and $\mathcal{B}_N$ determines the appropriate discrete boundary conditions by approximating the boundary operator, $\mathcal{B}$. We assume that the semi-discrete approximation is a consistent approximation of the continuous problem. A time-differencing scheme, where the boundary conditions are enforced exactly at the boundary points, may then be expressed as

$$\begin{array}{l} \mathbf{q}^{n+1} = K_N(\Delta t,\ \mathcal{L}_N)\mathbf{q}^n \ , \\ \mathcal{B}_N\mathbf{q}^{n+1} = 0 \ . \end{array}$$

Here $\mathbf{q}^n$ signifies the solution vector at time-step n. Thus, for strong stability we must require

$$|K_N(\Delta t,\ \mathcal{L}_N)| < 1 \ .$$

However, employing the penalty method changes the time-stepping scheme to be

$$\mathbf{q}^{n+1} = K_N(\Delta t\ , \mathcal{L}_N - \tau\mathcal{B}_N)\mathbf{q}^n,$$

and strong stability is ensured if

$$|K_N(\Delta t\ , \mathcal{L}_N - \tau\mathcal{B}_N)| < 1 \ ,$$

explaining why the CFL condition depends strongly on the correct choice of the penalty parameter.

In the following analysis we consider explicit Runge-Kutta time-stepping methods, which, for time independent operators, may be expressed as

$$K_N^p(\Delta t, \mathcal{L}_N) = \sum_{i=0}^{p} \frac{1}{i!}(\Delta t\mathcal{L}_N)^i \ ,$$

where p is the order of the scheme. We have for simplicity assumed that the boundary conditions are included in the operators. Assuming $\mathcal{L}_N = \mathcal{S}_N\Lambda_N\mathcal{S}_N^{-1}$, where $|\mathcal{S}_N|$ and $|\mathcal{S}_N^{-1}|$ are bounded independently of N, strong stability of the Runge-Kutta schemes is obtained if

$$|K_N^p(\Delta t,\ \mathcal{L}_N)| = \mathcal{S}_N\left|\sum_{i=0}^{p}\frac{1}{i!}(\Delta t\Lambda_N)^i\right|\mathcal{S}_N^{-1} = \left|\sum_{i=0}^{p}\frac{1}{i!}(\Delta t\Lambda_N)^i\right| < 1 \ .$$

TABLE 2

Scaling constants for the advection operator. The proper boundary conditions are of Dirichlet type (D).

Advection Operator		C_L		C_C	
$\lambda \neq 0$ $\varepsilon = 0$		3rd RK	4th RK	3rd RK	4th RK
D	Exact BC	21	35	27	32
	Penalty BC	10	17	10	11

TABLE 3

Scaling constants for the diffusion operator. Results are given for possible combinations of Dirichlet (D), Neumann (N), and Robin (R) boundary conditions.

Diffusion Operator		C_L		C_C	
$\lambda = 0$ $\varepsilon > 0$		3rd RK	4th RK	3rd RK	4th RK
D-D/D-N/D-R	Exact BC	99	109	53	58
	Penalty BC	81	123	56	84
N-R	Exact BC	99	109	53	58
	Penalty BC	130	135	91	96
R-R	Exact BC	99	109	53	58
	Penalty BC	130	141	93	97

Hence, the problem is reduced to finding the eigenvalue spectrum of the operator $\mathcal{L}_N$ and choosing Δt accordingly.

In the present study we consider the linear advection-diffusion operator

$$\mathcal{L} = \lambda \frac{\partial}{\partial x} + \varepsilon \frac{\partial^2}{\partial x^2} ,$$

with the Robin boundary condition operators

$$\mathcal{B}^- = \alpha - \varepsilon\beta \frac{\partial}{\partial x} \quad , \quad \mathcal{B}^+ = \gamma + \varepsilon\delta \frac{\partial}{\partial x} \quad .$$

The boundary conditions for the exact method are enforced through the operator as described in [18].

In order to compare time-step restrictions found for the two different approaches, we now define the two CFL-like constants, C_L and C_C, as

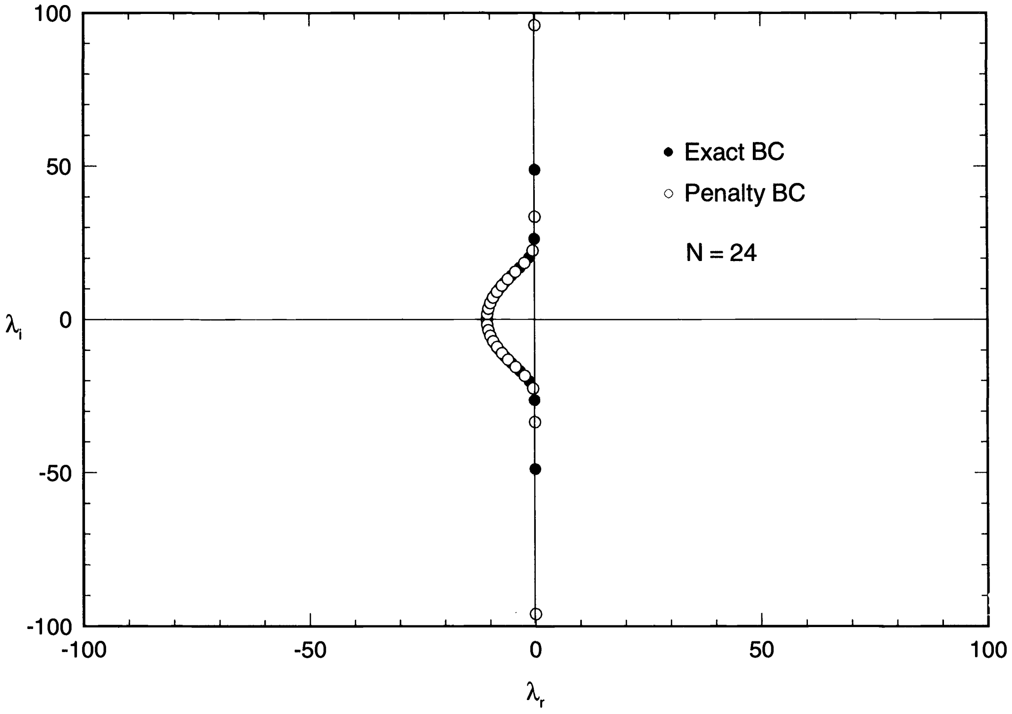
$$\Delta t_L \leq \frac{C_L}{\lambda N(N+1) + \varepsilon N^2(N+1)^2} \quad , \quad \Delta t_C \leq \frac{C_C}{\lambda N^2 + \varepsilon N^4} \quad ,$$

where the subscripts refer to Legendre (L) and Chebyshev (C) operators, respectively. These constants are determined by solving the eigenvalue problem and calculating the maximum Δt which ensures stability and, hence, supplies an upper bound on the time-step.

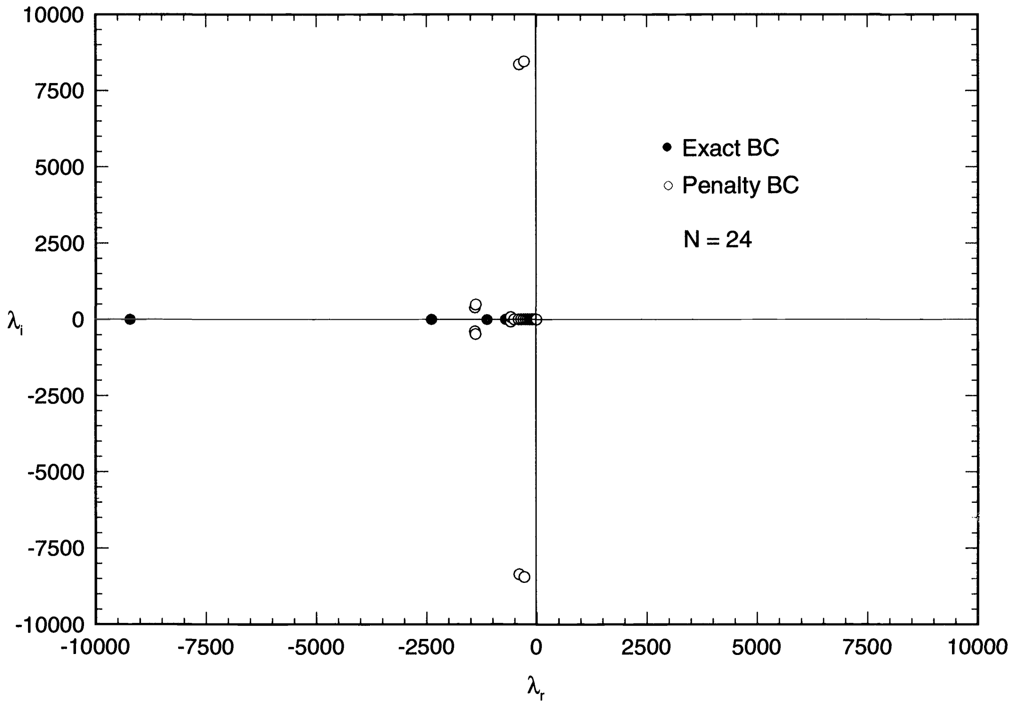
Tables 2 and 3 show the calculated values of C_L and C_C for the advection and the diffusion operator. The results are the same for the full advection-diffusion operator as for the diffusion operator, provided $\mathrm{Re}_N \ll 1$, and are therefore omitted.

It is clear from Table 2 that using the penalty method to enforce boundary conditions on purely advective problems results in a significant reduction of the maximum allowable time-step. However, more importantly, Table 3 shows that for problems where the diffusion operator dominates the eigenvalue spectrum, the penalty method may allow for increasing the time-step as much as 50%. The effect is most pronounced when using a fourth-order Runge-Kutta method for time-stepping a Chebyshev collocation scheme.

In order to explain the results in Tables 2 and 3, we compare in Fig. 2 the spectrum of the Legendre collocation advection (Fig. 2a) and diffusion (Fig. 2b) operators when enforcing Dirichlet boundary conditions through the exact method and the penalty method.



(a)



(b)

FIG. 2. *Eigenvalue spectrum ($\lambda = \lambda_r + i\,\lambda_i$) for the Legendre advection operator* (a) *and the Legendre diffusion operator* (b) *as obtained by using exact boundary conditions* (●) *and the penalty method* (○).

For the advection operator (Fig. 2a) we observe that the effect of the penalty method is to introduce an extreme, purely imaginary, complex conjugate eigenvalue-pair, which dominates the spectrum and consequently determines the maximum allowable time-step. This results in the decreased CFL number observed in Table 2.

The effect on the diffusion operator is more complicated and depends strongly on the value of the penalty parameter. As proved by Gottlieb and Lustman [17], the diffusion operator with exact Robin boundary conditions has a real, negative, and distinct eigenvalue spectrum. This property is preserved if a sufficiently large value of τ is used in the penalty method. However, by decreasing the penalty parameter the two dominating eigenvalues split into two pairs of complex conjugate eigenvalues, which move toward the imaginary axis as τ is decreased. In Fig. 2b we show the eigenvalue spectrum for the optimal choice of τ. The important observation to make is that moduli of these new eigenvalues are smaller than the original extreme negative real eigenvalue. Additionally, since the dominating eigenvalue now is complex, it clearly becomes advantageous to use the fourth-order Runge-Kutta method because of the increased extension of the stability region along the imaginary axis as compared to the third-order Runge-Kutta method. Thus, we conclude that for diffusion-dominated problems, the penalty method may allow for a significant increase in the maximum CFL number when applied in conjunction with the fourth-order Runge-Kutta method. This is true for Legendre as well as Chebyshev collocation methods.

The validity of this conclusion is, however, strongly dependent on the proper choice of the penalty parameter. The values derived in the previous section do indeed ensure asymptotic stability, but result in a significant reduction in the maximum allowable CFL number. Fortunately the limits of the penalty parameters are based on a conservative energy estimate and, consequently, are not very strict. In what follows we give the penalty parameters used to obtain the results given in Tables 2 and 3. These values result in a stable scheme as long as the problem is purely advective or $\mathrm{Re}_N \ll 1$, and allow in most cases for a significant increase in the time-step.

Legendre collocation methods.

1. Dirichlet boundary conditions.

$$\tau = \frac{|\lambda|}{4}N(N+1) + \frac{\varepsilon}{64}N^2(N+1)^2 \ .$$

2. Neumann boundary conditions.

$$\tau = \frac{N(N+1)}{8} \ .$$

3. Robin boundary conditions.

$$\tau = \frac{\tau^-_{\alpha,\beta}}{4} \ .$$

Chebyshev collocation methods.

1. Dirichlet boundary conditions.

$$\tau = \frac{|\lambda|}{2}N^2 + \frac{\varepsilon}{50}N^4 \ .$$

2. Neumann boundary conditions.

$$\tau = \frac{N^2}{8} \ .$$

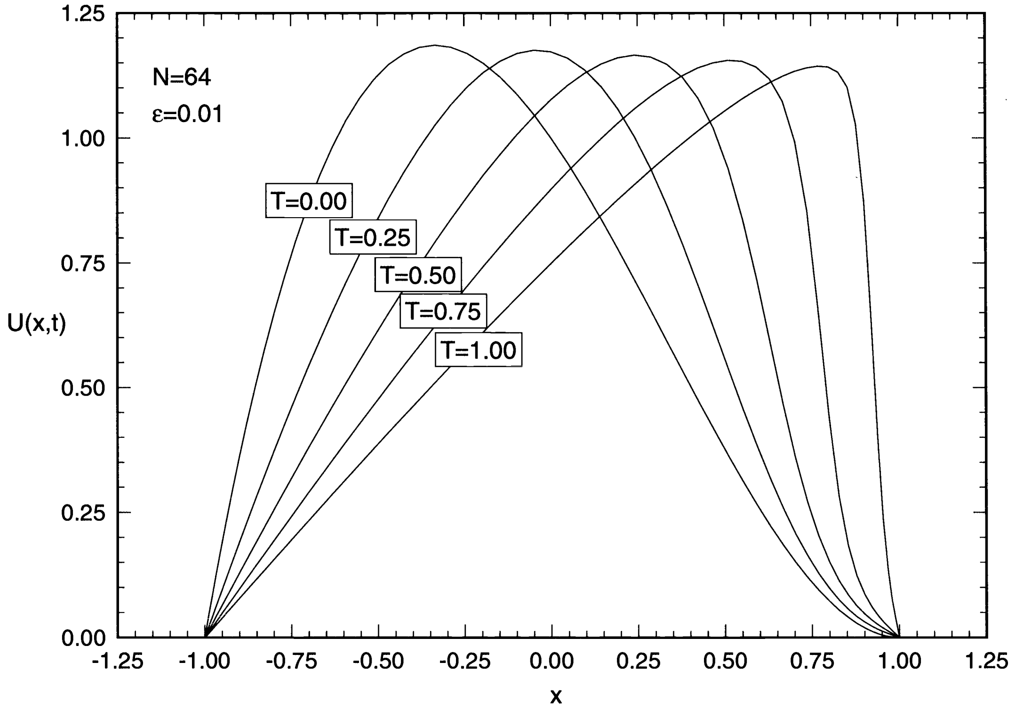


FIG. 3. *Temporal evolution of the Burgers equation with initial conditions given by* (12).

3. Robin boundary conditions.

$$\tau = \frac{\tau^{-}_{\alpha,\beta}}{4} \text{ with } \kappa = \frac{\alpha N^2}{\beta} .$$

Note that the only difference between the parameter values quoted here and those found in Lemma 3.2 is a factor of 1/4 on those terms related to the diffusion operator. This reduction is found to lead to optimal time-step restrictions.

We would like to stress the importance of choosing the appropriate value of the penalty parameter. It is our experience, that this is best done by deriving the theoretical value of this parameter through an analysis similar to that done in §3.1. This leads to a parameter which scales correctly with the resolution and other significant parameters. If the time-step restriction is dominated by a viscous time-scale, it is very likely that the theoretical estimate leads to severe time-step restrictions. However, the theoretical value may often be decreased considerably, and good results may be obtained after only a few tests. As we have seen for the Burgers equation, decreasing the penalty parameter four times leads to acceptable CFL restrictions. We are not aware of any systematic way to determine the optimal factor by which the theoretical value can be decreased, but it may usually be determined by trial and error through a few tests.

To conclude our study we have solved the Burgers equation, (3), with initial condition

$$U(x,0) = (1-x)(1-x^2) \tag{12}$$

and homogeneous Dirichlet boundary conditions. A typical temporal evolution is shown in Fig. 3. In Table 4 we show the maximum CFL number resulting in a stable scheme. This result confirms that the results from the linear analysis carry over to the scalar nonlinear problem.

TABLE 4
Maximum allowable CFL number obtained from direct numerical simulation of the Burgers equation.

	Legendre		Chebyshev	
	3rd RK	4th RK	3rd RK	4th RK
Exact BC	3.50	4.00	4.25	4.50
Penalty BC	3.00	4.50	4.75	6.75

5. The compressible Navier–Stokes equations. In the present section, we obtain energy estimates for the solution of the three-dimensional compressible Navier-Stokes equations given in conservation form. Additionally, we derive open boundary conditions taking into account the full stress-tensor, and prove well-posedness for the continuous problem. The derivations follow the approach introduced in [8, 9]. The main differences are that we develop the theory for the conservation form of the Navier-Stokes equations and that we include the off-diagonal terms of the stress-tensor in the full derivations. In the second part of this section we continue by showing how to apply the boundary conditions and prove asymptotic stability of the semi-discrete scheme.

Consider now the nondimensionalized, compressible Navier-Stokes equations given in conservation form

$$\frac{\partial \mathbf{q}}{\partial t}+\frac{\partial \mathbf{F}}{\partial x}+\frac{\partial \mathbf{G}}{\partial y}+\frac{\partial \mathbf{H}}{\partial z}=\frac{1}{\mathrm{Re}_{\mathrm{ref}}}\left(\frac{\partial \mathbf{F}_\nu}{\partial x}+\frac{\partial \mathbf{G}_\nu}{\partial y}+\frac{\partial \mathbf{H}_\nu}{\partial z}\right) , \tag{13}$$

with $\mathbf{x} \in \Omega = [-1, 1]^3$. The state vector, $\mathbf{q}$, and the inviscid flux vectors are given as

$$\mathbf{q}=\begin{bmatrix}\rho\\ \rho u\\ \rho v\\ \rho w\\ E\end{bmatrix} , \quad \mathbf{F}=\begin{bmatrix}\rho u\\ \rho u^2+p\\ \rho uv\\ \rho uw\\ (E+p)u\end{bmatrix} , \quad \mathbf{G}=\begin{bmatrix}\rho v\\ \rho uv\\ \rho v^2+p\\ \rho vw\\ (E+p)v\end{bmatrix} , \quad \mathbf{H}=\begin{bmatrix}\rho w\\ \rho uw\\ \rho vw\\ \rho w^2+p\\ (E+p)w\end{bmatrix} .$$

Here ρ is the density, u, v, w are the three Cartesian velocity components, E is the total energy, and p is the pressure. In the remaining part of this paper we will use (x, y, z) and (x_1, x_2, x_3) interchangeably to denote the spatial coordinates. The total energy

$$E=\rho\left(T+\frac{1}{2}\left(u^2+v^2+w^2\right)\right) ,$$

and the pressure are related through the ideal gas law

$$p=(\gamma-1)\rho T ,$$

where T is the temperature field and $\gamma = c_p/c_v$ is the ratio between the heat capacities at constant pressure (c_p) and volume (c_v), respectively, and is assumed constant.

The viscous flux vectors are given as

$$\mathbf{F}_\nu=\begin{bmatrix}0\\ \tau_{xx}\\ \tau_{yx}\\ \tau_{zx}\\ \tau_{xx}u+\tau_{yx}v+\tau_{zx}w+\frac{\gamma k}{\mathrm{Pr}}\frac{\partial T}{\partial x}\end{bmatrix} , \quad \mathbf{G}_\nu=\begin{bmatrix}0\\ \tau_{xy}\\ \tau_{yy}\\ \tau_{zy}\\ \tau_{xy}u+\tau_{yy}v+\tau_{zy}w+\frac{\gamma k}{\mathrm{Pr}}\frac{\partial T}{\partial y}\end{bmatrix} ,$$

$$\mathbf{H}_\nu=\begin{bmatrix}0\\ \tau_{xz}\\ \tau_{yz}\\ \tau_{zz}\\ \tau_{xz}u+\tau_{yz}v+\tau_{zz}w+\frac{\gamma k}{\mathrm{Pr}}\frac{\partial T}{\partial z}\end{bmatrix} .$$

Considering only Newtonian fluids, the stress-tensor elements are given as

$$\begin{aligned}
\tau_{xx} &= 2\mu\frac{\partial u}{\partial x} + \lambda\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right), &\quad \tau_{xy} = \tau_{yx} &= \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right),\\
\tau_{yy} &= 2\mu\frac{\partial v}{\partial y} + \lambda\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right), &\quad \tau_{yz} = \tau_{zy} &= \mu\left(\frac{\partial w}{\partial y} + \frac{\partial v}{\partial z}\right),\\
\tau_{zz} &= 2\mu\frac{\partial w}{\partial z} + \lambda\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right), &\quad \tau_{xz} = \tau_{zx} &= \mu\left(\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}\right),
\end{aligned}$$

where μ is the dynamic viscosity, λ is the bulk viscosity, and k is the coefficient of thermal conductivity.

The equations are normalized using the reference values, $u_{\text{ref}} = u_0$, $\rho_{\text{ref}} = \rho_0$, $p_{\text{ref}} = \rho_0 u_0^2$, $T_{\text{ref}} = u_0^2/c_v$, and a reference length L, where (ρ_0, u_0) is some uniform state, e.g., the ambient free-stream conditions of the flow. This gives a Reynolds number as $\text{Re} = \rho_0 u_0 L/\mu_0$ and a Prandtl number as $\text{Pr} = c_p\mu_0/k_0$. Note that the Reynolds number in (13), Re_{ref}, based on the reference values, in general is different from Re. In the remaining part of the paper we shall refer to the latter as the Reynolds number unless clarification is deemed necessary. With this normalization we need to specify the Mach number, M, the Reynolds number, Re, the length scale, L, and a dimensional temperature, T_0.

5.1. Well-posedness and open boundary conditions for the continuous problem. Consider the linearized, constant coefficient form of (13). The viscous fluxes are split as

$$\mathbf{F}_v = \mathbf{F}_P + \mathbf{F}_M^y + \mathbf{F}_M^z = \begin{bmatrix} 0 \\ (\lambda + 2\mu)\frac{\partial u}{\partial x} \\ \mu\frac{\partial v}{\partial x} \\ \mu\frac{\partial w}{\partial x} \\ (\lambda + 2\mu)u\frac{\partial u}{\partial x} + \mu v\frac{\partial v}{\partial x} + \mu w\frac{\partial w}{\partial x} + \frac{\gamma k}{\text{Pr}}\frac{\partial T}{\partial x} \end{bmatrix}$$

$$+ \begin{bmatrix} 0 \\ \lambda\frac{\partial v}{\partial y} \\ \mu\frac{\partial u}{\partial y} \\ 0 \\ \lambda u\frac{\partial v}{\partial y} + \mu v\frac{\partial u}{\partial y} \end{bmatrix} + \begin{bmatrix} 0 \\ \lambda\frac{\partial w}{\partial z} \\ 0 \\ \mu\frac{\partial u}{\partial z} \\ \lambda u\frac{\partial w}{\partial z} + \mu w\frac{\partial u}{\partial z} \end{bmatrix},$$

$$\mathbf{G}_v = \mathbf{G}_P + \mathbf{G}_M^x + \mathbf{G}_M^z = \begin{bmatrix} 0 \\ \mu\frac{\partial u}{\partial y} \\ (\lambda + 2\mu)\frac{\partial v}{\partial y} \\ \mu\frac{\partial w}{\partial y} \\ \mu u\frac{\partial u}{\partial y} + (\lambda + 2\mu)v\frac{\partial v}{\partial y} + \mu w\frac{\partial w}{\partial y} + \frac{\gamma k}{\text{Pr}}\frac{\partial T}{\partial y} \end{bmatrix}$$

$$+ \begin{bmatrix} 0 \\ \mu\frac{\partial v}{\partial x} \\ \lambda\frac{\partial u}{\partial x} \\ 0 \\ \lambda v\frac{\partial u}{\partial x} + \mu u\frac{\partial v}{\partial x} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \lambda\frac{\partial w}{\partial z} \\ \mu\frac{\partial v}{\partial z} \\ \lambda v\frac{\partial w}{\partial z} + \mu w\frac{\partial v}{\partial z} \end{bmatrix},$$

$$\mathbf{H}_v = \mathbf{H}_P + \mathbf{H}_M^x + \mathbf{H}_M^y = \begin{bmatrix} 0 \\ \mu\frac{\partial u}{\partial z} \\ \mu\frac{\partial v}{\partial z} \\ (\lambda + 2\mu)\frac{\partial w}{\partial z} \\ \mu u\frac{\partial u}{\partial z} + \mu v\frac{\partial v}{\partial z} + (\lambda + 2\mu) w\frac{\partial w}{\partial z} + \frac{\gamma k}{\mathrm{Pr}}\frac{\partial T}{\partial z} \end{bmatrix}$$

$$+ \begin{bmatrix} 0 \\ \mu\frac{\partial w}{\partial x} \\ 0 \\ \lambda\frac{\partial u}{\partial x} \\ \lambda w\frac{\partial u}{\partial x} + \mu u\frac{\partial w}{\partial x} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \mu\frac{\partial w}{\partial y} \\ \lambda\frac{\partial v}{\partial y} \\ \lambda w\frac{\partial v}{\partial y} + \mu v\frac{\partial w}{\partial y} \end{bmatrix} .$$

Introducing the transformation Jacobians

$$\mathcal{A}_1 = \frac{\partial \mathbf{F}}{\partial \mathbf{q}} \ , \ \mathcal{A}_2 = \frac{\partial \mathbf{G}}{\partial \mathbf{q}} \ , \ \mathcal{A}_3 = \frac{\partial \mathbf{H}}{\partial \mathbf{q}} \ ,$$

$$\mathcal{B}_{11} = \frac{\partial \mathbf{F}_P}{\partial \mathbf{q}_x} \ , \ \mathcal{B}_{22} = \frac{\partial \mathbf{G}_P}{\partial \mathbf{q}_y} \ , \ \mathcal{B}_{33} = \frac{\partial \mathbf{H}_P}{\partial \mathbf{q}_z} \ ,$$

$$\mathcal{B}_{12} = \left(\frac{\partial \mathbf{F}_M^y}{\partial \mathbf{q}_y} + \frac{\partial \mathbf{G}_M^x}{\partial \mathbf{q}_x}\right) \ , \ \mathcal{B}_{23} = \left(\frac{\partial \mathbf{G}_M^z}{\partial \mathbf{q}_z} + \frac{\partial \mathbf{H}_M^y}{\partial \mathbf{q}_y}\right) \ , \ \mathcal{B}_{13} = \left(\frac{\partial \mathbf{F}_M^z}{\partial \mathbf{q}_z} + \frac{\partial \mathbf{H}_M^x}{\partial \mathbf{q}_x}\right) \ ,$$

allows us to write Navier-Stokes equations as

$$\frac{\partial \mathbf{q}}{\partial t} + \sum_{i=1}^{3} \mathcal{A}_i \frac{\partial \mathbf{q}}{\partial x_i} = \frac{1}{\mathrm{Re}_{\mathrm{ref}}} \sum_{i=1}^{3}\sum_{j=i}^{3} \mathcal{B}_{ij} \frac{\partial^2 \mathbf{q}}{\partial x_i\, \partial x_j} \ .$$

It is well known that the Navier-Stokes equations, although not of hyperbolic nature, support waves very similar to those encountered in the hyperbolic Euler equations. For hyperbolic systems, Gottlieb, Gunzburger, and Turkel [19] have shown that enforcing the boundary conditions through the characteristic variables of the system results in a stable approximation.

For Navier-Stokes equations, we linearize around a uniform state, $\mathbf{q}_0$, by fixing all the matrices. We transform into characteristic variables by diagonalizing $\mathcal{A}_1$ through a similarity transform $\Lambda = \mathcal{S}^{-1}\mathcal{A}_1\mathcal{S}$, where Λ is the eigenvalue matrix and $\mathcal{S}$ and $\mathcal{S}^{-1}$ are the matrices of right and left eigenvectors, respectively. These matrices are given in the Appendix. Applying this, the symmetrized, linearized set of equations transforms into

$$\mathcal{Q}^T\mathcal{Q}\frac{\partial \mathbf{R}}{\partial t} + \sum_{i=1}^{3} \mathcal{A}_i^s \frac{\partial \mathbf{R}}{\partial x_i} = \frac{1}{\mathrm{Re}_{\mathrm{ref}}} \sum_{i=1}^{3}\sum_{j=i}^{3} \mathcal{B}_{ij}^s \frac{\partial^2 \mathbf{R}}{\partial x_i\, \partial x_j} \ , \tag{14}$$

where $\mathbf{R} = \mathcal{S}^{-1}\mathbf{q}$ are the characteristic variables. We have introduced a positive definite, symmetrizing diagonal matrix, $\mathcal{Q}^T\mathcal{Q}$, given as

$$\mathcal{Q}^T\mathcal{Q} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & \frac{2c_0^2}{\gamma-1} & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} ,$$

where $c_0 = \sqrt{\gamma p_0/\rho_0}$ is the uniform state sound speed. Additionally we define the symmetrized matrices

$$\mathcal{A}_i^s = \mathcal{Q}^T\mathcal{Q}\mathcal{S}^{-1}\mathcal{A}_i\mathcal{S} \ , \quad \mathcal{B}_{ij}^s = \mathcal{Q}^T\mathcal{Q}\mathcal{S}^{-1}\mathcal{B}_{ij}\mathcal{S} \ .$$

The explicit forms of the symmetric matrices are given in the Appendix. The characteristic variables, $\mathbf{R} = [R_1,\ R_2,\ R_3,\ R_4,\ R_5]^T$, are given as

$$\mathbf{R} = \begin{bmatrix} \rho u - u_0\rho + \frac{\gamma-1}{c_0}\left(E + \frac{1}{2}(u_0^2+v_0^2+w_0^2)\rho - u_0\rho u - v_0\rho v - w_0\rho w\right) \\ \rho v - v_0\rho \\ \rho - \frac{\gamma-1}{c_0^2}\left(E + \frac{1}{2}\rho(u_0^2+v_0^2+w_0^2) - u_0\rho u - v_0\rho v - w_0\rho w\right) \\ \rho w - w_0\rho \\ -(\rho u - u_0\rho) + \frac{\gamma-1}{c_0}\left(E + \frac{1}{2}\rho(u_0^2+v_0^2+w_0^2) - u_0\rho u - v_0\rho v - w_0\rho w\right) \end{bmatrix} .$$

We are now ready to state the following lemma.

LEMMA 5.1. *Assume there exists a solution, $\mathbf{q}$, which is periodic or held at a constant value at the y- and z-boundary. If the boundary conditions in the x-direction are given such that*

$$\forall (y,z) \in \Omega_y \times \Omega_z \ : \ -\frac{1}{2}\left[\mathbf{R}^T\mathcal{A}_1^s\mathbf{R} - \frac{2}{\mathrm{Re}_{\mathrm{ref}}}\sum_{j=1}^{3}\mathbf{R}^T\mathcal{B}_{1j}^s\frac{\partial\mathbf{R}}{\partial x_j}\right]_{x=-1}^{1} \le 0 \ ,$$

and the fluid properties are constrained by

$$\mu_0 \ge 0 \ , \quad \lambda_0 \le 0 \ , \quad \lambda_0 + \mu_0 \ge 0 \ , \quad \frac{\gamma k_0}{\mathrm{Pr}} \ge 0 \ , \quad \gamma \ge 1 \ ,$$

then (14) *constitutes a well-posed problem and the solution is bounded as*

$$\frac{1}{2}\frac{d}{dt}\|\mathcal{Q}\mathbf{R}\|^2 \le -\frac{1}{\mathrm{Re}_{\mathrm{ref}}}\int_\Omega\left(\sum_{i=1}^{3}\sum_{j=i}^{3}\frac{\partial\mathbf{R}^T}{\partial x_i}\mathcal{B}_{ij}^s\frac{\partial\mathbf{R}}{\partial x_j}\right)d\Omega \le 0 \ .$$

Proof. Construct the energy integral as

$$\begin{aligned}\frac{1}{2}\frac{d}{dt}\|\mathcal{Q}\mathbf{R}\|^2 &= \int_\Omega\left(-\sum_{i=1}^{3}\mathbf{R}^T\mathcal{A}_i^s\frac{\partial\mathbf{R}}{\partial x_i} + \frac{1}{\mathrm{Re}_{\mathrm{ref}}}\sum_{i=1}^{3}\sum_{j=i}^{3}\mathbf{R}^T\mathcal{B}_{ij}^s\frac{\partial^2\mathbf{R}}{\partial x_i\partial x_j}\right)d\Omega \\ &= \int_{\Omega_y}\int_{\Omega_z} -\frac{1}{2}\left[\mathbf{R}^T\mathcal{A}_1^s\mathbf{R} - \frac{2}{\mathrm{Re}_{\mathrm{ref}}}\sum_{j=1}^{3}\mathbf{R}^T\mathcal{B}_{1j}^s\frac{\partial\mathbf{R}}{\partial x_j}\right]_{x=-1}^{1} dydz \\ &\quad - \frac{1}{\mathrm{Re}_{\mathrm{ref}}}\int_\Omega\left(\sum_{i=1}^{3}\sum_{j=i}^{3}\frac{\partial\mathbf{R}^T}{\partial x_i}\mathcal{B}_{ij}^s\frac{\partial\mathbf{R}}{\partial x_j}\right)d\Omega \ ,\end{aligned}$$

where $\Omega = \Omega_x \times \Omega_y \times \Omega_z$. In deriving this expression, we use partial integration and assume the solution to be periodic or held at a constant value along the y- and z-boundaries, i.e., contributions from these boundaries cancel. This is not a severe restriction, as this assumption is valid for a large variety of situations where open boundary conditions are applied.

It is evident that if we can prove

$$\text{(15)} \qquad \frac{1}{\mathrm{Re}_{\mathrm{ref}}}\int_\Omega\left(\sum_{i=1}^{3}\sum_{j=i}^{3}\frac{\partial\mathbf{R}^T}{\partial x_i}\mathcal{B}_{ij}^s\frac{\partial\mathbf{R}}{\partial x_j}\right)d\Omega \ge 0 \ ,$$

then well-posedness may be ensured by properly constructing the boundary operator at the x-boundary.

Since the matrices, $\mathcal{B}^s_{ij}$, are all symmetric, (15) may be rewritten in a block-quadratic form as

$$\frac{1}{2\mathrm{Re}_{\mathrm{ref}}}\int_\Omega \tilde{\mathbf{R}}^T \mathcal{H}^s \tilde{\mathbf{R}}\, d\Omega \geq 0 \ ,$$

where we have introduced

$$\tilde{\mathbf{R}} = \left[\frac{\partial \mathbf{R}}{\partial x}, \frac{\partial \mathbf{R}}{\partial y}, \frac{\partial \mathbf{R}}{\partial z}\right]^T \ , \ \mathcal{H}^s = \begin{bmatrix} 2\mathcal{B}^s_{11} & \mathcal{B}^s_{12} & \mathcal{B}^s_{13} \\ \mathcal{B}^s_{12} & 2\mathcal{B}^s_{22} & \mathcal{B}^s_{23} \\ \mathcal{B}^s_{13} & \mathcal{B}^s_{23} & 2\mathcal{B}^s_{33} \end{bmatrix} \ .$$

We observe that $\mathcal{H}^s$ is a 15×15 symmetric block matrix, ensuring that the eigenvalue spectrum, $\rho(\mathcal{H}^s)$, is real. Hence, if $\mathcal{H}^s$ is positive semi-definite, (15) is obeyed. The eigenvalue spectrum, $\rho(\mathcal{H}^s)$, may be found to be

$$\begin{aligned}
\rho_1 &= \rho_2 = \rho_3 = 0 \ , \\
\rho_4 &= 2(\mu_0 - \lambda_0) \ , \\
\rho_5 &= \rho_6 = 2(\lambda_0 + 3\mu_0) \ , \\
\rho_7 &= \rho_8 = 3\mu_0 - \sqrt{\mu_0^2 + 2(\mu_0 + \lambda_0)^2} \ , \\
\rho_9 &= \rho_{10} = 3\mu_0 + \sqrt{\mu_0^2 + 2(\mu_0 + \lambda_0)^2} \ , \\
\rho_{11} &= 7\mu_0 + 4\lambda_0 - \sqrt{\mu_0^2 + 4(\mu_0 + \lambda_0)(3\mu_0 + 2\lambda_0)} \ , \\
\rho_{12} &= 7\mu_0 + 4\lambda_0 + \sqrt{\mu_0^2 + 4(\mu_0 + \lambda_0)(3\mu_0 + 2\lambda_0)} \ , \\
\rho_{13} &= \rho_{14} = \rho_{15} = \left(\frac{2c_0^2}{(\gamma - 1)^2} + 1\right)\frac{2(\gamma - 1)k_0}{\mathrm{Pr}} \ .
\end{aligned}$$

Here subscript "0" signifies the parameter values in the uniform state around which we have linearized. For most real fluids under nonextreme conditions, it is true that μ_0 is positive, λ_0 is negative, and the following relationship is obeyed [20]:

$$\frac{\gamma\mu_0}{\mathrm{Pr}} \geq \lambda_0 + 2\mu_0 \geq \mu_0 \ . \tag{16}$$

A simple investigation of the eigenvalues reveals that $\mathcal{H}^s$ is positive semi-definite under these conditions. Thus, (15) is true provided

$$\mu_0 \geq 0 \ , \quad \lambda_0 \leq 0 \ , \quad \lambda_0 + \mu_0 \geq 0 \ , \quad \frac{\gamma k_0}{\mathrm{Pr}} \geq 0 \ , \quad \gamma \geq 1 \ .$$

These conditions are only natural as discussed in [21]. In fact, if they are not obeyed, Navier-Stokes equations violate the second law of thermodynamics.

We now obtain that well-posedness is ensured under the additional condition

$$\forall (y,z) \in \Omega_y \times \Omega_z \quad -\frac{1}{2}\left[\mathbf{R}^T \mathcal{A}^s_1 \mathbf{R} - \frac{2}{\mathrm{Re}_{\mathrm{ref}}}\sum_{j=1}^{3} \mathbf{R}^T \mathcal{B}^s_{1j}\frac{\partial \mathbf{R}}{\partial x_j}\right]_{x=-1}^{1} \leq 0 \ ,$$

with the solution being bounded as

$$\frac{1}{2}\frac{d}{dt}\|\mathcal{Q}\mathbf{R}\|^2 \leq -\frac{1}{\mathrm{Re}_{\mathrm{ref}}}\int_\Omega \left(\sum_{i=1}^{3}\sum_{j=i}^{3}\frac{\partial \mathbf{R}^T}{\partial x_i}\mathcal{B}^s_{ij}\frac{\partial \mathbf{R}}{\partial x_j}\right) d\Omega \leq 0 \ ,$$

where $\mathcal{Q}\mathbf{R} = \mathcal{Q}\mathcal{S}^{-1}\mathbf{q}$. □

As stated in Lemma 5.1, appropriate boundary conditions at the x-boundary have to obey

$$-\frac{1}{2}\left[\mathbf{R}^T\mathcal{A}^s_1\mathbf{R} - \frac{2}{\mathrm{Re}_{\mathrm{ref}}}\sum_{j=1}^{3}\mathbf{R}^T\mathcal{B}^s_{1j}\frac{\partial \mathbf{R}}{\partial x_j}\right]_{-1}^{1} \leq 0 \ .$$

We now define

$$\mathcal{Q}^T\mathcal{Q}\mathbf{G} = \sum_{j=1}^{3}\mathcal{B}^s_{1j}\frac{\partial \mathbf{R}}{\partial x_j} \ ,$$

where

$$\begin{aligned}
G_1 &= \frac{k_0(\gamma-1)}{2\rho_0\mathrm{Pr}}\frac{\partial\zeta_1}{\partial x} + \frac{\lambda_0+2\mu_0}{2\rho_0}\frac{\partial\zeta_2}{\partial x} + \frac{\lambda_0+\mu_0}{\rho_0}\left(\frac{\partial R_2}{\partial y}+\frac{\partial R_4}{\partial z}\right) \ , \\
G_2 &= \frac{\mu_0}{\rho_0}\frac{\partial R_2}{\partial x} + \frac{\lambda_0+\mu_0}{2\rho_0}\frac{\partial\zeta_2}{\partial y} \ , \\
G_3 &= -\frac{k_0(\gamma-1)}{2\rho_0 c_0\mathrm{Pr}}\frac{\partial\zeta_1}{\partial x} \ , \\
G_4 &= \frac{\mu_0}{\rho_0}\frac{\partial R_4}{\partial x} + \frac{\lambda_0+\mu_0}{2\rho_0}\frac{\partial\zeta_2}{\partial z} \ , \\
G_5 &= \frac{k_0(\gamma-1)}{2\rho_0\mathrm{Pr}}\frac{\partial\zeta_1}{\partial x} - \frac{\lambda_0+2\mu_0}{2\rho_0}\frac{\partial\zeta_2}{\partial x} - \frac{\lambda_0+\mu_0}{\rho_0}\left(\frac{\partial R_2}{\partial y}+\frac{\partial R_4}{\partial z}\right) \ ,
\end{aligned} \tag{17}$$

where we, for simplicity, have introduced

$$\zeta_1 = R_1 + R_5 - \frac{2c_0}{\gamma-1}R_3 \ , \ \zeta_2 = R_1 - R_5 \ .$$

Some physical meaning can be given to the components of $\mathbf{G}$. The three components G_1, G_3, and G_5 account for effects caused by the normal heat flux and stress, whereas the remaining two, G_2 and G_4, are a consequence of tangential stress at the boundary.

This formulation allows for rewriting the constraint on the boundary contribution as

$$-\frac{1}{2}\mathcal{Q}^T\left[\mathbf{R}^T\Lambda\mathbf{R} - \frac{2}{\mathrm{Re}_{\mathrm{ref}}}\mathbf{R}^T\mathbf{G}\right]_{-1}^{1}\mathcal{Q} \leq 0 \ ,$$

where Λ is the diagonal eigenvalue matrix obtained from the similarity transform. Because $\mathcal{Q}^T\mathcal{Q}$ is positive definite, this inequality may be reformulated as

$$-\frac{1}{2}\left[\sum_{i=1}^{5}\lambda_i^{-1}\left(\left(|\lambda_i|R_i - \varepsilon\frac{|\lambda_i|}{\lambda_i}G_i\right)^2 - (\varepsilon G_i)^2\right)\right]_{-1}^{1} \leq 0 \ , \tag{18}$$

where λ_i are the wave speeds by which the characteristic variables are advected, as given by the diagonal elements of Λ, and we have introduced $\varepsilon = \mathrm{Re}_{\mathrm{ref}}{}^{-1}$. This formulation makes it straightforward to devise inflow-outflow boundary conditions, which are maximal dissipative and ensure well-posedness of the complete problem.

We note in particular that this formulation takes into account the off-diagonal terms of the stress-tensor, which are neglected in most previous work [8-10]. These terms may be of importance if the artificial boundary is introduced into a strongly vortical region of the flow, e.g., a wake flow behind a blunt body.

Inflow boundary conditions. At $x = -1$, (18) becomes

$$\frac{1}{2}\sum_{i=1}^{5}\lambda_i^{-1}\left(\left(|\lambda_i|R_i - \varepsilon\frac{|\lambda_i|}{\lambda_i}G_i\right)^2 - (\varepsilon G_i)^2\right) \le 0 \ .$$

Subsonic inflow: $\lambda_1 > 0$, $\lambda_2 > 0$, $\lambda_3 > 0$, $\lambda_4 > 0$, $\lambda_5 < 0$.

$$\begin{aligned} \lambda_1 R_1 - \varepsilon G_1 &= 0 \ , \\ \lambda_2 R_2 - \varepsilon G_2 &= 0 \ , \\ \lambda_3 R_3 - \varepsilon G_3 &= 0 \ , \\ \lambda_4 R_4 - \varepsilon G_4 &= 0 \ , \\ \varepsilon G_5 &= 0 \ . \end{aligned} \tag{19}$$

Supersonic inflow: $\lambda_1 > 0$, $\lambda_2 > 0$, $\lambda_3 > 0$, $\lambda_4 > 0$, $\lambda_5 > 0$.

$$\begin{aligned} \lambda_1 R_1 - \varepsilon G_1 &= 0 \ , \\ \lambda_2 R_2 - \varepsilon G_2 &= 0 \ , \\ \lambda_3 R_3 - \varepsilon G_3 &= 0 \ , \\ \lambda_4 R_4 - \varepsilon G_4 &= 0 \ , \\ \lambda_5 R_5 - \varepsilon G_5 &= 0 \ . \end{aligned} \tag{20}$$

Outflow boundary conditions. At $x = 1$, (18) becomes

$$\frac{1}{2}\sum_{i=1}^{5}-\lambda_i^{-1}\left(\left(|\lambda_i|R_i - \varepsilon\frac{|\lambda_i|}{\lambda_i}G_i\right)^2 - (\varepsilon G_i)^2\right) \le 0 \ .$$

Subsonic outflow: $\lambda_1 > 0$, $\lambda_2 > 0$, $\lambda_3 > 0$, $\lambda_4 > 0$, $\lambda_5 < 0$.

$$\begin{aligned} \varepsilon G_2 &= 0 \ , \\ \varepsilon G_3 &= 0 \ , \\ \varepsilon G_4 &= 0 \ , \\ |\lambda_5|R_5 + \varepsilon G_5 &= 0 \ . \end{aligned} \tag{21}$$

Supersonic outflow: $\lambda_1 > 0$, $\lambda_2 > 0$, $\lambda_3 > 0$, $\lambda_4 > 0$, $\lambda_5 > 0$.

$$\begin{aligned} \varepsilon G_1 &= 0 \ , \\ \varepsilon G_2 &= 0 \ , \\ \varepsilon G_3 &= 0 \ , \\ \varepsilon G_4 &= 0 \ . \end{aligned} \quad \text{or} \quad \begin{aligned} \varepsilon G_2 &= 0 \ , \\ \varepsilon G_3 &= 0 \ , \\ \varepsilon G_4 &= 0 \ , \\ \varepsilon G_5 &= 0 \ . \end{aligned} \tag{22}$$

We note that for both types of outflow boundary conditions, it is only necessary to specify four conditions, since $\varepsilon G_3 = 0 \Rightarrow \varepsilon G_1 = -\varepsilon G_5$. Due to the special structure of **G** we also observe that adding an extra condition on εG_1 at the outflow does not place extra conditions on the solution, since such a condition is redundant. This observation will be used later.

It was shown by Strikwerda [22] that the proper number of boundary conditions for an incomplete, parabolic system as the compressible Navier-Stokes equations is 5 in the inflow region and 4 in the outflow region. Our result clearly conforms with that.

We also note that in the limit of infinite Reynolds number, these boundary conditions converge uniformly toward the well-known characteristic boundary conditions for the compressible Euler equations [23]. This property is important in order to avoid weak boundary layers of the order $\exp(-x/\varepsilon)$ (see [8]).

5.2. The semi-discrete scheme. Following the line of thought that led to the asymptotically stable scheme for Burgers equation, we now propose a penalty method for enforcing open boundary conditions to a Legendre collocation approximation of the compressible Navier-Stokes equations

$$\frac{\partial \mathbf{q}}{\partial t}+\frac{\partial \mathbf{F}}{\partial x}+\frac{\partial \mathbf{G}}{\partial y}+\frac{\partial \mathbf{H}}{\partial z}=\frac{1}{\mathrm{Re}_{\mathrm{ref}}}\left(\frac{\partial \mathbf{F}_v}{\partial x}+\frac{\partial \mathbf{G}_v}{\partial y}+\frac{\partial \mathbf{H}_v}{\partial z}\right) \\ -\tau_1 \mathcal{Q}^-(x)\mathcal{S}\left(\mathcal{R}^-\left(\mathbf{R}-\mathcal{S}^{-1}\mathbf{g}_1(t)\right)-\frac{1}{\mathrm{Re}_{\mathrm{ref}}}\mathcal{G}^-\mathbf{G}\right) \\ -\tau_2 \mathcal{Q}^+(x)\mathcal{S}\left(\mathcal{R}^+\left(\mathbf{R}-\mathcal{S}^{-1}\mathbf{g}_2(t)\right)+\frac{1}{\mathrm{Re}_{\mathrm{ref}}}\mathcal{G}^+\mathbf{G}\right) . \tag{23}$$

Here $\mathcal{Q}^-(x)$ and $\mathcal{Q}^+(x)$ are given by (9) and $\mathcal{S}$ is the right eigenvector matrix as given in the Appendix. The boundary conditions for the state vector are given through the two vectors, $\mathbf{g}_1(t)$ and $\mathbf{g}_2(t)$, which we for convenience assume to represent a uniform state outside of the computational domain. The four matrices $\mathcal{R}^-$, $\mathcal{R}^+$, $\mathcal{G}^-$, and $\mathcal{G}^+$ are chosen to construct the appropriate boundary operator as derived in the previous section. Hence, we have for the inflow region

$$\mathcal{R}^-=\begin{bmatrix}\lambda_1&0&0&0&0\\0&\lambda_2&0&0&0\\0&0&\lambda_3&0&0\\0&0&0&\lambda_4&0\\0&0&0&0&\alpha\lambda_5\end{bmatrix}, \quad \mathcal{G}^-=\begin{bmatrix}1&0&0&0&0\\0&1&0&0&0\\0&0&1&0&0\\0&0&0&1&0\\0&0&0&0&1\end{bmatrix},$$

where $\alpha=0$ for subsonic conditions and $\alpha=1$ for supersonic conditions. Likewise we define

$$\mathcal{R}^+=\begin{bmatrix}0&0&0&0&0\\0&0&0&0&0\\0&0&0&0&0\\0&0&0&0&0\\0&0&0&0&\beta|\lambda_5|\end{bmatrix}, \quad \mathcal{G}^+=\begin{bmatrix}0&0&0&0&0\\0&1&0&0&0\\0&0&1&0&0\\0&0&0&1&0\\0&0&0&0&1\end{bmatrix},$$

where $\beta=1$ for subsonic conditions and $\beta=0$ for supersonic outflow conditions.

We have to choose τ_1 and τ_2 such that the semi-discrete scheme is asymptotically stable. The proper choice is stated in the following lemma.

LEMMA 5.2. *Assume there exists a solution,* $\mathbf{q}$, *which is periodic or held at a constant value at the* y*- and* z*-boundary, and that the fluid properties of the uniform state,* $\mathbf{q}_0$, *are constrained by*

$$\mu_0 \geq 0 \ , \quad \lambda_0 \leq 0 \ , \quad \lambda_0+\mu_0 \geq 0 \ , \quad \frac{\gamma k_0}{\mathrm{Pr}} \geq 0 \ , \quad \gamma \geq 1 \ ,$$

and related as

$$\frac{\gamma\mu_0}{\mathrm{Pr}} \geq \lambda_0+2\mu_0 \geq \mu_0 \ .$$

The linearized, constant coefficient version of the scheme given by (23) *is asymptotically stable at the* inflow *if*

$$\frac{1}{\omega\kappa}\left(1+\kappa+\sqrt{1+\kappa}\right) \geq \tau_1 \geq \frac{1}{\omega\kappa}\left(1+\kappa-\sqrt{1+\kappa}\right) .$$

Here

$$\kappa = \frac{\varepsilon}{2\omega}\frac{\gamma k_0}{\mathrm{Pr}\rho_0 u_0} .$$

This result is independent of whether the inflow is subsonic or supersonic.

For supersonic outflow

$$\frac{1}{\omega}\left(1+\sqrt{\frac{1}{\kappa}}\right) \geq \tau_2 \geq \frac{1}{\omega}\left(1-\sqrt{\frac{1}{\kappa}}\right) .$$

For subsonic outflow

$$\frac{1}{\omega\kappa}\left(1+\kappa+\sqrt{1+\kappa}\right) \geq \tau_2 \geq \frac{1}{\omega\kappa}\left(1+\kappa-\sqrt{1+\kappa}\right) .$$

The solution to (23) *is bounded in the form*

$$\frac{1}{2}\frac{d}{dt}\|\mathcal{Q}\mathbf{R}\|_N^2 \leq -\frac{1}{\mathrm{Re}_{\mathrm{ref}}}\sum_{k=1}^{N-1}\omega_k\int_{\Omega_y}\int_{\Omega_z}\left[\sum_{i=1}^{3}\sum_{j=i}^{3}\frac{\partial\mathbf{R}^T}{\partial x_i}\mathcal{B}^s_{ij}\frac{\partial\mathbf{R}}{\partial x_j}\right]_{x=x_k} dy\,dz \leq 0 .$$

Proof. Write (23) in its symmetrized, linearized, constant coefficient version

$$\begin{aligned}\mathcal{Q}^T\mathcal{Q}\frac{\partial\mathbf{R}}{\partial t}+\sum_{i=1}^{3}\mathcal{A}^s_i\frac{\partial\mathbf{R}}{\partial x_i} =&\frac{1}{\mathrm{Re}_{\mathrm{ref}}}\sum_{i=1}^{3}\sum_{j=i}^{3}\mathcal{B}^s_{ij}\frac{\partial^2\mathbf{R}}{\partial x_i\,\partial x_j}\\ &-\tau_1 Q^-(x)\left(\mathcal{Q}^T\mathcal{Q}\mathcal{R}^-\mathbf{R}-\frac{1}{\mathrm{Re}_{\mathrm{ref}}}\mathcal{Q}^T\mathcal{Q}\mathcal{G}^-\mathbf{G}\right)\\ &-\tau_2 Q^+(x)\left(\mathcal{Q}^T\mathcal{Q}\mathcal{R}^+\mathbf{R}+\frac{1}{\mathrm{Re}_{\mathrm{ref}}}\mathcal{Q}^T\mathcal{Q}\mathcal{G}^+\mathbf{G}\right) ,\end{aligned}$$

where, without loss of generality, we have assumed homogeneous boundary conditions. We construct the energy integral, apply the Gauss-Lobatto quadrature rule, and use partial integration to obtain

$$\begin{aligned}(24)\quad \frac{1}{2}\frac{d}{dt}\|\mathcal{Q}\mathbf{R}\|_N^2=&\int_{\Omega_y}\int_{\Omega_z}-\frac{1}{2}\left[\mathbf{R}^T\mathcal{A}^s_1\mathbf{R}-2\varepsilon\sum_{j=1}^{3}\mathbf{R}^T\mathcal{B}^s_{1j}\frac{\partial\mathbf{R}}{\partial x_j}\right]_{x=-1}^{1} dy\,dz\\ &-\varepsilon\int_{\Omega}\left(\sum_{i=1}^{3}\sum_{j=i}^{3}\frac{\partial\mathbf{R}^T}{\partial x_i}\mathcal{B}^s_{ij}\frac{\partial\mathbf{R}}{\partial x_j}\right)d\Omega\\ &-\tau_1\omega\int_{\Omega_y}\int_{\Omega_z}\left[\mathbf{R}^T\mathcal{Q}^T\mathcal{Q}\mathcal{R}^-\mathbf{R}-\varepsilon\mathbf{R}^T\mathcal{G}^-\sum_{j=1}^{3}\mathcal{B}^s_{1j}\frac{\partial\mathbf{R}}{\partial x_j}\right]_{x=-1} dy\,dz\\ &-\tau_2\omega\int_{\Omega_y}\int_{\Omega_z}\left[\mathbf{R}^T\mathcal{Q}^T\mathcal{Q}\mathcal{R}^+\mathbf{R}+\varepsilon\mathbf{R}^T\mathcal{G}^+\sum_{j=1}^{3}\mathcal{B}^s_{1j}\frac{\partial\mathbf{R}}{\partial x_j}\right]_{x=1} dy\,dz ,\end{aligned}$$

where we have used the assumption about periodicity or constant value at the y- and z-boundary. Additionally, we have introduced $\varepsilon = \mathrm{Re_{ref}}^{-1}$ and ω, which is the Legendre weight at the endpoints and applied the definition

$$\mathcal{Q}^T \mathcal{Q} \mathbf{G} = \sum_{j=1}^{3} \mathcal{B}^s_{1j} \frac{\partial \mathbf{R}}{\partial x_j} \ .$$

Using the Gauss-Lobatto quadrature rule allows us to write

$$\text{(25)} \quad \int_\Omega \left(\sum_{i=1}^{3} \sum_{j=i}^{3} \frac{\partial \mathbf{R}^T}{\partial x_i} \mathcal{B}^s_{ij} \frac{\partial \mathbf{R}}{\partial x_j} \right) d\Omega = \int_{\Omega_y} \int_{\Omega_z} \left(\sum_{k=1}^{N-1} \omega_k \left[\sum_{i=1}^{3} \sum_{j=i}^{3} \frac{\partial \mathbf{R}^T}{\partial x_i} \mathcal{B}^s_{ij} \frac{\partial \mathbf{R}}{\partial x_j} \right]_{x=x_k} \right. \\ + \omega \left[\sum_{i=1}^{3} \sum_{j=i}^{3} \frac{\partial \mathbf{R}^T}{\partial x_i} \mathcal{B}^s_{ij} \frac{\partial \mathbf{R}}{\partial x_j} \right]_{x=-1} \\ \left. + \ \omega \left[\sum_{i=1}^{3} \sum_{j=i}^{3} \frac{\partial \mathbf{R}^T}{\partial x_i} \mathcal{B}^s_{ij} \frac{\partial \mathbf{R}}{\partial x_j} \right]_{x=1} \right) dy\, dz \geq 0 \ .$$

Here x_k signifies the Legendre-Gauss-Lobatto collocation points. The inequality follows from the analysis done in the proof of Lemma 5.1 and is ensured provided the fluid properties are constrained by

$$\mu_0 \geq 0 \ , \quad \lambda_0 \leq 0 \ , \quad \lambda_0 + \mu_0 \geq 0 \ , \quad \frac{\gamma k_0}{\mathrm{Pr}} \geq 0 \ , \quad \gamma \geq 1 \ .$$

It was shown by Abarbanel and Gottlieb [20] that if a scheme is stable without the contributions from the off-diagonal stress-tensor terms, then it will remain so even if the these terms are included. This is a consequence of the general relation

$$\frac{\gamma \mu_0}{\mathrm{Pr}} \geq \lambda_0 + 2\mu_0 \geq \mu_0 \ ,$$

which roughly gives the relation between the eigenvalues of the normal stress-tensor elements and the off-diagonal elements. Thus, it is sufficient to prove stability in the absence of the off-diagonal contributions.

The penalty parameters, τ_1 and τ_2, must be chosen such that the boundary term of the energy integral does not destroy the stability of the Cauchy problem. We treat the two boundary contributions separately.

Inflow condition. The contribution of the boundary term at the inflow ($x = -1$) follows from combining (24) and (25) and neglecting the off-diagonal contributions to obtain

$$\mathbf{R}^T \left(\frac{1}{2} \mathcal{A}^s_1 - \tau_1 \omega \mathcal{Q}^T \mathcal{Q} \mathcal{R}^- \right) \mathbf{R} - \varepsilon \mathbf{R}^T \left(\mathcal{I} - \tau_1 \omega \mathcal{G}^- \right) \mathcal{B}^s_{11} \frac{\partial \mathbf{R}}{\partial x_1} - \varepsilon \omega \sum_{i=1}^{3} \frac{\partial \mathbf{R}^T}{\partial x_i} \mathcal{B}^s_{ii} \frac{\partial \mathbf{R}}{\partial x_i} \leq 0 \ ,$$

where $\mathcal{I}$ is the identity matrix.

First we note that

$$-\varepsilon \omega \frac{\partial \mathbf{R}^T}{\partial x_2} \mathcal{B}^s_{22} \frac{\partial \mathbf{R}}{\partial x_2} - \varepsilon \omega \frac{\partial \mathbf{R}^T}{\partial x_3} \mathcal{B}^s_{33} \frac{\partial \mathbf{R}}{\partial x_3} \leq 0 \ ,$$

since $\mathcal{B}^s_{22}$ and $\mathcal{B}^s_{33}$ are positive semi-definite with an eigenvalue spectrum given as

$$\rho_1(\mathcal{B}^s_{22}) = \rho_1(\mathcal{B}^s_{33}) = 0 \ , \qquad \rho_2(\mathcal{B}^s_{22}) = \rho_2(\mathcal{B}^s_{33}) = \frac{\mu_0}{\rho_0} \ ,$$

$$\rho_3(\mathcal{B}^s_{22}) = \rho_3(\mathcal{B}^s_{33}) = 2\frac{\mu_0}{\rho_0} \ , \qquad \rho_4(\mathcal{B}^s_{22}) = \rho_4(\mathcal{B}^s_{33}) = 2\frac{\lambda_0 + 2\mu_0}{\rho_0} \ ,$$

$$\rho_5(\mathcal{B}^s_{22}) = \left(\frac{2c_0^2}{(\gamma-1)^2} + 1 \right) \frac{(\gamma-1)k_0}{\rho_0 \mathrm{Pr}} \ , \quad \rho_5(\mathcal{B}^s_{33}) = \left(\frac{2c_0^2}{(\gamma-1)^2} + 1 \right) \frac{(\gamma-1)k_0}{\rho_0 \mathrm{Pr}} \ .$$

Since all matrices are symmetric, the remaining part of the constraint may be expressed in block-quadratic form as

$$\tilde{\mathbf{R}}^T \mathcal{H}^- \tilde{\mathbf{R}} \le 0 \ ,$$

where

$$\tilde{\mathbf{R}} = \left[\mathbf{R}, \frac{\partial \mathbf{R}}{\partial x} \right]^T \ , \quad \mathcal{H}^- = \frac{1}{2} \begin{bmatrix} \mathcal{A}_1^s - 2\tau_1 \omega \mathcal{Q}^T \mathcal{Q} \mathcal{R}^- & -\varepsilon(1-\tau_1\omega)\mathcal{B}_{11}^s \\ -\varepsilon(1-\tau_1\omega)\mathcal{B}_{11}^s & -2\varepsilon\omega\mathcal{B}_{11}^s \end{bmatrix} \ ,$$

where we have used $\mathcal{G}^- = \mathcal{I}$. $\mathcal{H}^-$ is a 10 × 10 symmetric block-matrix. Similar to the approach applied in §3.2, we have transformed the problem of stability into proving that $\mathcal{H}^-$, for a suitable value of τ_1, is negative semi-definite. The eigenvalue-spectrum, $\rho(\mathcal{H}^-)$, can be found by doing an LU-decomposition. Since $\mathcal{H}^-$ is symmetric, the eigenvalues appear as $\rho_i(\mathcal{H}^-) = U_{ii}$, i.e., the diagonal elements of the upper triangular matrix.

We will not give the general form of the eigenvalues here, since they are rather complicated. However, straightforward but very lengthy algebra shows that all eigenvalues are negative if τ_1 is chosen such that

$$\frac{1}{\omega\kappa}\left(1+\kappa+\sqrt{1+\kappa}\right) \ge \tau_1 \ge \frac{1}{\omega\kappa}\left(1+\kappa-\sqrt{1+\kappa}\right) \ ,$$

where

$$\kappa = \frac{\varepsilon}{2\omega} \frac{\gamma k_0}{\mathrm{Pr}\rho_0 u_0} \ .$$

This result is independent of whether the inflow is subsonic or supersonic.

Outflow condition. Neglecting the contribution from the off-diagonal terms yields a criteria for stability at the outflow ($x = 1$)

$$-\mathbf{R}^T \left(\frac{1}{2}\mathcal{A}_1^s + \tau_2\omega\mathcal{Q}^T\mathcal{Q}\mathcal{R}^+ \right) \mathbf{R} + \varepsilon \mathbf{R}^T \left(\mathcal{I} - \tau_2\omega\mathcal{G}^+ \right) \mathcal{B}_{11}^s \frac{\partial \mathbf{R}}{\partial x_1} - \varepsilon\omega \sum_{i=1}^{3} \frac{\partial \mathbf{R}^T}{\partial x_i} \mathcal{B}_{ii}^s \frac{\partial \mathbf{R}}{\partial x_i} \le 0 \ .$$

Similar to the approach followed in the previous part of the proof, we see that the contributions from $\mathcal{B}_{22}^s$ and $\mathcal{B}_{33}^s$ are always negative and independently ensure stability.

We now rewrite the remaining part of the condition at the outflow in block-quadratic form:

$$\tilde{\mathbf{R}}^T \mathcal{H}^+ \tilde{\mathbf{R}} \le 0 \ ,$$

where

$$\mathcal{H}^+ = \frac{1}{2} \begin{bmatrix} -\mathcal{A}_1^s - 2\tau_2 \omega \mathcal{Q}^T \mathcal{Q} \mathcal{R}^+ & \varepsilon(1-\tau_2\omega)\mathcal{B}_{11}^s \\ \varepsilon(1-\tau_1\omega)\mathcal{B}_{11}^s & -2\varepsilon\omega\mathcal{B}_{11}^s \end{bmatrix} \ .$$

To form $\mathcal{H}^+$ we have assumed $\mathcal{G}^+ = \mathcal{I}$. The additional boundary condition introduced by this replacement is redundant, as discussed in §5.1, and, hence, no extra restrictions are put on the system by this approach. The eigenvalue spectrum, $\rho(\mathcal{H}^+)$, may again be found through an LU-decomposition. We state here only the bounds on τ_2 that ensure negative semi-definiteness of $\mathcal{H}^+$ for supersonic outflow

$$\frac{1}{\omega}\left(1+\sqrt{\frac{1}{\kappa}}\right) \ge \tau_2 \ge \frac{1}{\omega}\left(1-\sqrt{\frac{1}{\kappa}}\right) \ .$$

For subsonic outflow the bounds become

$$\frac{1}{\omega\kappa}\left(1+\kappa+\sqrt{1+\kappa}\right) \ge \tau_2 \ge \frac{1}{\omega\kappa}\left(1+\kappa-\sqrt{1+\kappa}\right) \; .$$

Combining (24) and (25), we obtain a bound for the growth of the solution

$$\frac{1}{2}\frac{d}{dt}\|\mathcal{Q}\mathbf{R}\|_N^2 \le -\frac{1}{\mathrm{Re}_{\mathrm{ref}}}\sum_{k=1}^{N-1}\omega_k\int_{\Omega_y}\int_{\Omega_z}\left[\sum_{i=1}^{3}\sum_{j=i}^{3}\frac{\partial \mathbf{R}^T}{\partial x_i}\mathcal{B}^s_{ij}\frac{\partial \mathbf{R}}{\partial x_j}\right]_{x=x_k} dy\,dz \le 0 \; . \qquad \square$$

We wish to emphasize that the bounds on τ_1 and τ_2 given in Lemma 5.2 remain valid in the limit when the Reynolds number approaches infinity. This is easily realized by expanding the bounds for $\varepsilon \ll 1$ to obtain

$$\infty > \tau_1 \ge \frac{1}{2\omega} + \varepsilon\frac{1}{8\omega}\kappa$$

in the inflow region and

$$\infty > \tau_2 > -\infty \quad , \quad \infty > \tau_2 \ge \frac{1}{2\omega} + \varepsilon\frac{1}{8\omega}\kappa$$

for supersonic and subsonic outflow, respectively. The linearized, constant coefficient version of the Euler equations may be transformed into 5 independent hyperbolic equations for which we should expect the bounds on the penalty parameters to be given by the results in §3.1.1. We observe that the bounds given above converge uniformly to the expected values in the limit of vanishing viscosity and, thus, the scheme remains stable. The observation that no bounds are necessary on τ_2 for supersonic outflow simply reflects the fact that no boundary conditions are required for the Euler equations at such a boundary.

5.3. Numerical tests. The proof given in the previous section is only strictly valid for the linearized, constant coefficient version of Navier-Stokes equations. To validate the results and show that it carries over to full nonlinear Navier-Stokes equations, we have implemented the scheme in an existing spectral code (see [24] for details), originally developed for studying two-dimensional compressible flow around an infinitely long circular cylinder.

For the spatial approximation scheme we used a standard Chebyshev-Fourier collocation scheme in polar coordinates, (r, θ), with a third-order Runge-Kutta method for time-stepping.

The new scheme is simple to implement in existing codes, because we only need to apply a correction of the flux of the state vector at the boundary. Following the scheme, given by (23), we need to derive the two vectors **R** and **G**. The characteristic variables are given as

$$\begin{aligned} R_1 &= (m_r - \rho u_r) + \frac{p}{c_0} \; , \\ R_2 &= m_\theta - \rho u_\theta \; , \\ R_3 &= \rho - \frac{p}{c_0^2} \; , \\ R_4 &= -(m_r - \rho u_r) + \frac{p}{c_0} \; , \end{aligned}$$

where c_0 is the uniform state sound speed.

We have for convenience introduced

$$u_r = u_0\hat{k}_1 + v_0\hat{k}_2 \; , \quad u_\theta = u_0\hat{k}_2 - v_0\hat{k}_1 \; ,$$

which are the radial and azimuthal velocity components, respectively, of the uniform state, and

$$m_r = m_u \hat{k}_1 + m_v \hat{k}_2 \ , \quad m_\theta = m_u \hat{k}_2 - m_v \hat{k}_1 \ ,$$

which are the radial and azimuthal components of the momentum of the flow field. Here $\mathbf{k} = (\hat{k}_1, \hat{k}_2)$ signifies an outward pointing normal-vector at the boundary. The linearized pressure, p, is given as

$$p = (\gamma - 1)\left[E + \frac{1}{2}\rho(u_0^2 + v_0^2) - u_0 m_u - v_0 m_v\right] \ .$$

The eigenvalues corresponding to the characteristic functions and determining the direction and propagation velocity of the characteristic waves are

$$\lambda_1 = u_r + c_0 \ , \ \lambda_2 = \lambda_3 = u_r \ , \ \ \lambda_4 = u_r - c_0 \ .$$

Following the approach outlined in the previous section, we have likewise derived the viscous correction vector, $\mathbf{G}$, at the outer boundary as

$$G_1 = \frac{1}{\rho_0 r}\left[\frac{(\gamma-1)k_0}{2\mathrm{Pr}}\frac{\partial(r\zeta_1)}{\partial r} + \frac{2}{3}\mu_0\frac{\partial(r\zeta_2)}{\partial r} - \frac{1}{3}\mu_0\frac{\partial R_2}{\partial\theta}\right] \ ,$$
$$G_2 = \frac{\mu_0}{\rho_0 r}\left[\frac{\partial(rR_2)}{\partial r} - \frac{1}{6}\frac{\partial\zeta_2}{\partial\theta}\right] \ ,$$
$$G_3 = -\frac{(\gamma-1)k_0}{\mathrm{Pr}}\frac{1}{2c_0\rho_0}\frac{1}{r}\frac{\partial(r\zeta_1)}{\partial r} \ ,$$
$$G_4 = \frac{1}{\rho_0 r}\left[\frac{(\gamma-1)k_0}{2\mathrm{Pr}}\frac{\partial(r\zeta_1)}{\partial r} - \frac{2}{3}\mu_0\frac{\partial(r\zeta_2)}{\partial r} + \frac{1}{3}\mu_0\frac{\partial R_2}{\partial\theta}\right] \ ,$$

where again we have defined

$$\zeta_1 = R_1 + R_4 - \frac{2c_0}{\gamma - 1}R_2 \ , \ \ \zeta_2 = R_1 - R_4 \ ,$$

and applied Stokes hypothesis to obtain $\lambda = -\frac{2}{3}\mu$. We note that only two extra calculations of derivatives, $(\partial\rho/\partial r, \ \partial\rho/\partial\theta)$, are needed in order to form the two vectors, since the radial and azimuthal derivatives at the boundary of the remaining variables are calculated during evaluation of the interior dynamics when employing a global scheme. Thus, compared with evaluation of the flux, the computational requirement for enforcing the boundary conditions through this new method is negligible.

The boundary conditions are enforced at each intermediate time-step of the Runge-Kutta method. Simulations were done with a Reynolds number of 100, a Mach number of 0.4, a diameter (L) of the cylinder of 6.10 mm, and a reference temperature of 300°K. These parameters ensure that the flow field remains subsonic. The resolution was 96 Fourier modes, 72 Chebyshev modes and the radius (L_D) of the computational domain was 20 cylinder diameters.

As penalty parameters we used

$$\tau_1 = \tau_2 = \frac{N^2}{4\kappa}\frac{2}{L_D}(1 + \kappa - \sqrt{1+\kappa}) \ ,$$

where N is the number of Chebyshev modes, $2/L_D$ is a result of the radial mapping of L_D into $[-1, 1]$, and

$$\kappa = \frac{\varepsilon N^2}{2}\frac{\gamma k_0}{\mathrm{Pr}\rho_0|u_r|} \ ,$$

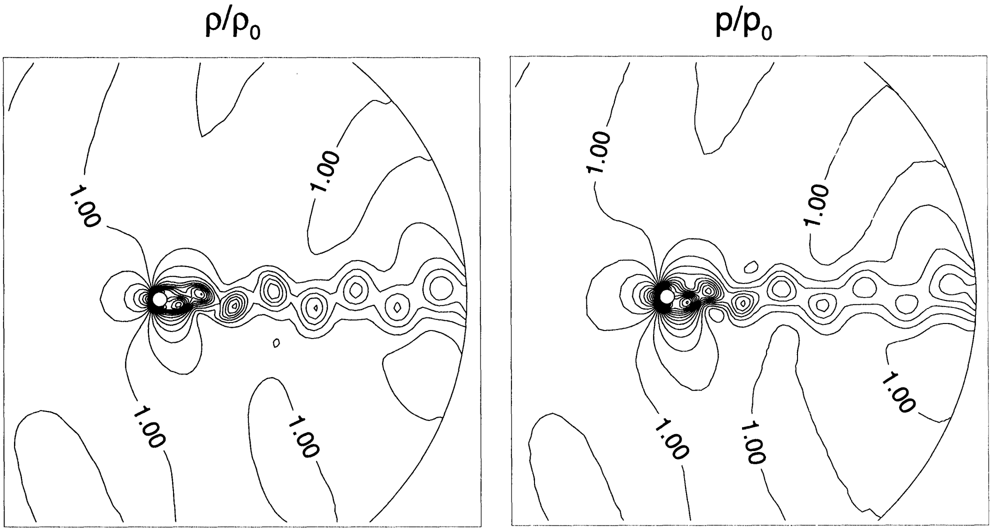


FIG. 4. *Contour plots of the normalized density,* ρ/ρ_0, *and the normalized pressure,* p/p_0, *at the nondimensional time* $T = 143.5$ *for a flow at* Re = 100, M = 0.4, $D = 6.10$ mm, *and* $T_0 = 300$°K.

This choice appears natural from the results stated in Lemma 5.2 and the experience gained in §4, indicating that for dissipative problems we may reduce the penalty parameter by a factor of 4 to obtain the optimal value of τ. With this choice of penalty parameters we were able to perform the simulations without any reduction in time-step as compared with the exact method of enforcing the boundary conditions. It should be mentioned that in the original code only characteristic boundary conditions for the Euler equations were enforced. Comparing with results discussed in §4, we observe that for third-order Runge-Kutta methods we should expect the two methods to impose almost equivalent time-step restrictions. This is confirmed by the simulations and shows that the results from the simple linear analysis carries over to the full nonlinear Navier-Stokes equations in this case.

In Fig. 4 we show contour plots of the normalized density and the pressure at $T = 143.5$, corresponding to approximately 23 shedding cycles. The von Karman vortex street is clearly demonstrated, and we observe that the boundary conditions at the outflow boundary affect the flow only slightly. The Strouhal number for the shredding frequency is found to be $St = 0.163$, which is in full accordance with experimental findings [25] and we observe no spurious frequencies or reflections from the artificial boundary back into the flow field (see [24] for a further discussion of this).

6. Concluding remarks. The purpose of the present paper has been twofold. The first goal has been to develop boundary conditions for wave-dominated problems, leading to well-posed total problems. It was argued that for smooth solutions and the class of operators we have considered here, it is sufficient to consider the problem of well-posedness for the locally linearized, constant coefficient version of the nonlinear initial-boundary value problem. Using this allowed the derivation of proper boundary conditions to the Burgers equation and to the three-dimensional, compressible Navier-Stokes equations, and these boundary conditions were shown to ensure well-posedness of the total problem. It should be stressed that the boundary conditions derived for the Navier-Stokes equations take into account all elements of the stress-tensor, and only very light assumptions were made to derive these. Additionally, they remain valid even in the limit of vanishing viscosity.

Having derived appropriate boundary conditions naturally leads to the question of how to enforce these in a discrete approximation of the problem. This has been the second, and main, contribution of the paper. Results [7] on the connection between stability of discrete and semi-discrete approximations suggest that it is sufficient to consider asymptotic stability for the semi-discrete approximation. We have only considered Legendre collocation methods here. This choice is dictated merely by a wish to obtain analytical results and we have indicated, by numerical tests, that all results carry over to Chebyshev collocation methods. The stability proofs for the semi-discrete approximations to the linearized, constant coefficient versions of the Burgers equation and the compressible Navier-Stokes equations are all completed by using the classical energy method. We emphasized that the proposed schemes remain stable even in the limit where the problems become purely hyperbolic.

The proposed penalty method changes the eigenvalue spectra of the discrete approximations of the operators considerably. In order to understand this, we performed a detailed investigation of the effect of the penalty method on the eigenvalue spectra of linear operators. It has been shown that the value of the penalty parameter, which is obtained from the theoretical analysis, often implies that the maximum allowable time-step compares unfavorably with that allowed for more traditional methods. However, we discussed in detail how to remedy this and showed that choosing the penalty parameter properly may in some cases allow for increasing the maximum time-step by as much as 50%. Although we are not aware of a systematic way of determining the optimal value of the penalty parameter, we do not see that as a significant disadvantage. Our experience tells that once the theoretical values of the penalty parameters are obtained, only a few tests are needed to obtain the optimal value. Additionally, this has to be done only once, and since only a few hundred time-steps are required to test whether the scheme is stable, we consider this an insignificant problem.

Most of the theoretical results, obtained for linearized, constant coefficient versions of the equations, are confirmed by numerical simulations of the full nonlinear equations. It is stressed that the proposed penalty method is very easy to implement in existing codes, which is an attractive feature.

Although all results and numerical simulations in this paper are obtained using spectral collocation methods, the main conclusions carry over to finite difference/finite element methods. The derivation of the proper boundary operators, for either the Burgers equation or for the compressible, Navier-Stokes equations, is obviously unaffected by the choice of the spatial approximation method. The proposed penalty method for enforcing the boundary conditions may be applied in exactly the same manner as discussed here when using alternative spatial discretization methods. The only difference is the value of the penalty parameter, which will depend strongly on the order of the method. Thus, applying another method requires one to derive this penalty parameter. This may be done by an approach equivalent to the one utilized here.

In a future paper [26], we will extend the penalty method developed here to include multidomain solutions of the compressible Navier-Stokes equations in general curvilinear coordinates.

Appendix: Symmetric matrices for the Navier-Stokes equations. Consider the linearized, constant coefficient compressible Navier-Stokes equations in conservation form given as

$$\frac{\partial \mathbf{q}}{\partial t} + \sum_{i=1}^{3} \mathcal{A}_i \frac{\partial \mathbf{q}}{\partial x_i} = \frac{1}{\mathrm{Re}_{\mathrm{ref}}} \sum_{i=1}^{3} \sum_{j=i}^{3} \mathcal{B}_{ij} \frac{\partial^2 \mathbf{q}}{\partial x_i \, \partial x_j} \quad .$$

The matrix, $\mathcal{A}_1$, diagonalizes under the similarity transform, $\Lambda = \mathcal{S}^{-1}\mathcal{A}_1\mathcal{S}$, where the right eigenvector matrix, $\mathcal{S}$, and the left eigenvector matrix, $\mathcal{S}^{-1}$, are given as

$$\mathcal{S} = \begin{bmatrix} \alpha & 0 & 1 & 0 & \alpha \\ \alpha(u+c) & 0 & u & 0 & \alpha(u-c) \\ \alpha v & 1 & v & 0 & \alpha v \\ \alpha w & 0 & w & 1 & \alpha w \\ \alpha(H+cu) & v & \frac{1}{2}c^2M^2 & w & \alpha(H-cu) \end{bmatrix},$$

$$\mathcal{S}^{-1} = \begin{bmatrix} \beta\left(\frac{1}{2}(\gamma-1)c^2M^2 - cu\right) & -\beta((\gamma-1)u-c) & -\beta(\gamma-1)v & -\beta(\gamma-1)w & \beta(\gamma-1) \\ -v & 0 & 1 & 0 & 0 \\ 1-\frac{1}{2}(\gamma-1)M^2 & \frac{\gamma-1}{c^2}u & \frac{\gamma-1}{c^2}v & \frac{\gamma-1}{c^2}w & -\frac{\gamma-1}{c^2} \\ -w & 0 & 0 & 1 & 0 \\ \beta\left(\frac{1}{2}(\gamma-1)c^2M^2 + cu\right) & -\beta((\gamma-1)u+c) & -\beta(\gamma-1)v & -\beta(\gamma-1)w & \beta(\gamma-1) \end{bmatrix}.$$

Here

$$\alpha = \frac{1}{2c}, \quad \beta = \frac{1}{c}.$$

Introducing this transformation into the Navier–Stokes equations yields

$$\mathcal{Q}^T\mathcal{Q}\frac{\partial \mathbf{R}}{\partial t} + \sum_{i=1}^{3} \mathcal{A}_i^s \frac{\partial \mathbf{R}}{\partial x_i} = \frac{1}{\mathrm{Re}_{\mathrm{ref}}} \sum_{i=1}^{3}\sum_{j=i}^{3} \mathcal{B}_{ij}^s \frac{\partial^2 \mathbf{R}}{\partial x_i\, \partial x_j},$$

where $\mathbf{R}$ are the characteristic variables and $\mathcal{Q}^T\mathcal{Q}$ is a positive definite, symmetrizing diagonal matrix.

The symmetrized matrices

$$\mathcal{A}_i^s = \mathcal{Q}^T\mathcal{Q}\mathcal{S}^{-1}\mathcal{A}_i\mathcal{S}, \quad \mathcal{B}_{ij}^s = \mathcal{Q}^T\mathcal{Q}\mathcal{S}^{-1}\mathcal{B}_{ij}\mathcal{S}$$

are given as

$$\mathcal{A}_1^s = \begin{bmatrix} u+c & 0 & 0 & 0 & 0 \\ 0 & 2u & 0 & 0 & 0 \\ 0 & 0 & \frac{2c^2}{\gamma-1}u & 0 & 0 \\ 0 & 0 & 0 & 2u & 0 \\ 0 & 0 & 0 & 0 & u-c \end{bmatrix}, \quad \mathcal{A}_2^s = \begin{bmatrix} v & c & 0 & 0 & 0 \\ c & 2v & 0 & 0 & c \\ 0 & 0 & \frac{2c^2}{\gamma-1}v & 0 & 0 \\ 0 & 0 & 0 & 2v & 0 \\ 0 & c & 0 & 0 & v \end{bmatrix},$$

$$\mathcal{A}_3^s = \begin{bmatrix} w & 0 & 0 & c & 0 \\ 0 & 2w & 0 & 0 & 0 \\ 0 & 0 & \frac{2c^2}{\gamma-1}w & 0 & 0 \\ c & 0 & 0 & 2w & c \\ 0 & 0 & 0 & c & w \end{bmatrix},$$

$$\mathcal{B}_{11}^s = \frac{1}{2\rho}\begin{bmatrix} (\lambda+2\mu)+\theta & 0 & -\frac{2c}{\gamma-1}\theta & 0 & -(\lambda+2\mu)+\theta \\ 0 & 4\mu & 0 & 0 & 0 \\ -\frac{2c}{\gamma-1}\theta & 0 & \frac{4c^2}{(\gamma-1)^2}\theta & 0 & -\frac{2c}{\gamma-1}\theta \\ 0 & 0 & 0 & 4\mu & 0 \\ -(\lambda+2\mu)+\theta & 0 & -\frac{2c}{\gamma-1}\theta & 0 & (\lambda+2\mu)+\theta \end{bmatrix},$$

$$\mathcal{B}_{22}^{s} = \frac{1}{2\rho}\begin{bmatrix} \mu+\theta & 0 & -\frac{2c}{\gamma-1}\theta & 0 & -\mu+\theta \\ 0 & 4(\lambda+2\mu) & 0 & 0 & 0 \\ -\frac{2c}{\gamma-1}\theta & 0 & \frac{4c^2}{(\gamma-1)^2}\theta & 0 & -\frac{2c}{\gamma-1}\theta \\ 0 & 0 & 0 & 4\mu & 0 \\ -\mu+\theta & 0 & -\frac{2c}{\gamma-1}\theta & 0 & \mu+\theta \end{bmatrix},$$

$$\mathcal{B}_{33}^{s} = \frac{1}{2\rho}\begin{bmatrix} \mu+\theta & 0 & -\frac{2c}{\gamma-1}\theta & 0 & -\mu+\theta \\ 0 & 4\mu & 0 & 0 & 0 \\ -\frac{2c}{\gamma-1}\theta & 0 & \frac{4c^2}{(\gamma-1)^2}\theta & 0 & -\frac{2c}{\gamma-1}\theta \\ 0 & 0 & 0 & 4(\lambda+2\mu) & 0 \\ -\mu+\theta & 0 & -\frac{2c}{\gamma-1}\theta & 0 & \mu+\theta \end{bmatrix}.$$

We have for convenience introduced

$$\theta = \frac{\gamma-1}{\gamma}\frac{\gamma k}{\mathrm{Pr}},$$

$$\mathcal{B}_{12}^{s} = \frac{\lambda+\mu}{\rho}\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \end{bmatrix}, \quad \mathcal{B}_{13}^{s} = \frac{\lambda+\mu}{\rho}\begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 \end{bmatrix},$$

$$\mathcal{B}_{23}^{s} = \frac{\lambda+\mu}{\rho}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Acknowledgment. The authors would like to thank Dr. W. S. Don and Dr. L. Dettori from Brown University for many useful discussions.

REFERENCES

[1] H. O. KREISS AND J. LORENZ, *Initial-Boundary Value Problems and the Navier–Stokes Equations*, Series in Pure and Applied Mathematics, Academic Press, San Diego, CA, 1989.

[2] A. HARTEN, *On the symmetric form of systems of conservation laws with entropy*, J. Comput. Phys., 49 (1983), pp. 151-164.

[3] D. FUNARO AND D. GOTTLIEB, *A new method of imposing boundary conditions in pseudospectral approximations of hyperbolic equations*, Math. Comp., 51 (1988), pp. 599-613.

[4] ———, *Convergence results for pseudospectral approximations of hyperbolic systems by a penalty-type boundary treatment*, Math. Comp., 57 (1991), pp. 585-596.

[5] M. CARPENTER, D. GOTTLIEB, AND S. ABARBANEL, *Time-stable boundary conditions for finite-difference schemes solving hyperbolic systems: Methodology and applications to high-order compact schemes*, ICASE Report no. 93-9, 1993.

[6] W. S. DON AND D. GOTTLIEB, *The Chebyshev-Legendre method: Implementing Legendre methods on Chebyshev points*, SIAM J. Numer. Anal., 31 (1994), pp. 1519-1534.

[7] H. O. KREISS AND L. WU, *On the stability definition of difference approximations for the initial boundary value problem*, Appl. Numer. Math., 12 (1993), pp. 213-227.

[8] B. GUSTAFSSON AND A. SUNDSTRÖM, *Incompletely parabolic problems in fluid dynamics*, SIAM J. Appl. Math., 35 (1978), pp. 343-357.

[9] J. OLIGER AND A. SUNDSTRÖM, *Theoretical and practical aspects of some initial boundary value problems in fluid dynamics*, SIAM J. Appl. Math., 35 (1978), pp. 419-446.

[10] J. NORDSTRÖM, *Artificial Boundary Conditions for the Navier–Stokes Equations*, Ph.D. Thesis, Uppsala University, Sweden, 1993.

[11] J. NORDSTRÖM, *The use of characteristic boundary conditions for the Navier–Stokes equations*, Comput. Fluids, 24 (1995), pp. 609-623.

[12] P. DUTT, *Stable boundary conditions and difference schemes for Navier–Stokes equations*, SIAM J. Numer. Anal., 25 (1988), pp. 245-267.

[13] L. HALPERN, *Artificial boundary conditions for incompletely parabolic perturbations of hyperbolic systems*, SIAM J. Math. Anal., 22 (1991), pp. 1256-1283.

[14] P. J. DAVIS AND P. RABINOWITZ, *Methods of Numerical Integration*. 2nd ed., Academic Press, San Diego, 1984.

[15] D. GOTTLIEB, M. Y. HUSSAINI, AND S. A. ORSZAG, *Introduction: Theory and applications of spectral methods*, in Spectral Methods for Partial Differential Equations, R. Voigt, D. Gottlieb, and M.Y. Hussaini, eds., SIAM, Philadelphia, 1984, pp. 1-54.

[16] D. FUNARO, *Polynomial Approximation of Differential Equations*, Springer-Verlag, New York, 1992.

[17] D. GOTTLIEB AND L. LUSTMAN, *The spectrum of the Chebyshev collocation operator for the heat equation*, SIAM J. Numer. Anal., 20 (1983), pp. 909-921.

[18] C. CANUTO, M. Y. HUSSAINI, A. QUARTERONI, AND T. A. ZANG, *Spectral Methods in Fluid Dynamics*, Springer Series in Computational Physics, Springer-Verlag, New York, 1988.

[19] D. GOTTLIEB, M. GUNZBURGER, AND E. TURKEL, *On numerical boundary treatment of hyperbolic systems for finite difference and finite element methods*, SIAM J. Numer. Anal., 19 (1982), pp. 671-682.

[20] S. ABARBANEL AND D. GOTTLIEB, *Optimal time splitting for two- and three-dimensional Navier–Stokes equations with mixed derivatives*, J. Comput. Phys., 41 (1981), pp. 1-33.

[21] T. J. R. HUGHES, L. P. FRANCA, AND M. MALLET, *A new finite element formulation for computational fluid dynamics: I. Symmetric forms of the compressible Euler and Navier–Stokes equations and the second law of thermodynamics*, Comput. Methods Appl. Mech. Engrg., 54 (1986), pp. 223-234.

[22] J. C. STRIKWERDA, *Initial boundary value problems for incompletely parabolic systems*, Comm. Pure Appl. Math., 30 (1977), pp. 797-822.

[23] K. W. THOMPSON, *Time-dependent boundary conditions for hyperbolic systems*, J. Comput. Phys., 68 (1987), pp. 1-29.

[24] W. S. DON AND D. GOTTLIEB, *Spectral simulation of an unsteady compressible flow past a circular cylinder*, Comput. Methods Appl. Mech. Engrg., 80 (1990), pp. 39-58.

[25] C. H. K. WILLIAMSON, *Oblique and parallel modes of vortex shedding in the wake of a circular cylinder at low Reynolds numbers*, J. Fluid Mech., 206 (1989), pp. 579-627.

[26] J. S. HESTHAVEN, *A stable penalty method for the compressible Navier–Stokes equations.* II. *One dimensional domain decomposition schemes*, SIAM J. Sci. Comput., 18 (1997), to appear.

SIAM J. Sci. Comput.
Vol. 17, No. 3, pp. 613-630, May 1996

003

SEMICIRCULANT SOLVERS AND BOUNDARY CORRECTIONS FOR FIRST-ORDER PARTIAL DIFFERENTIAL EQUATIONS*

SVERKER HOLMGREN† AND KURT OTTO†

Abstract. Fast solvers for systems of partial differential equations (PDEs) in two space dimensions are considered. The solvers are used as direct solution methods or as preconditioners for a conjugate gradient (CG)-like iteration. Mainly first-order PDEs are considered, but second-order terms may be included. Employing a semicirculant framework, PDEs with constant coefficients in one space direction and arbitrary boundary conditions are considered. A factorization of the inverse of the difference approximation matrix is described. This factorization is exploited to derive a direct solver, where the complexity for the first right-hand side is $\mathcal{O}(n^{3/2}\log_2 n)$, but only $\mathcal{O}(n\log_2 n)$ for subsequent right-hand sides. From the factorization an iteration resembling the Schur complement matrix method known from domain decomposition is also constructed. The eigendecomposition of the iteration matrix is investigated. The new solution methods are compared with iterations with semicirculant preconditioners and to Gaussian elimination. An application solving the time-dependent, almost incompressible Navier–Stokes equations is also studied.

Key words. first-order PDE, finite-difference discretizations, fast solvers, low-rank corrections

AMS subject classifications. 65N22, 65F10

1. Introduction.

We study the solution of time-dependent and time-independent PDEs on the unit square:

$$\tag{1}\begin{cases} \left(\dfrac{\partial \mathbf{u}}{\partial t}+\right) \mathcal{A}_1\dfrac{\partial \mathbf{u}}{\partial x_1}+\mathcal{B}_1\dfrac{\partial^2 \mathbf{u}}{\partial x_1^2}+\mathcal{A}_2\dfrac{\partial \mathbf{u}}{\partial x_2}+\mathcal{B}_2\dfrac{\partial^2 \mathbf{u}}{\partial x_2^2}=\mathbf{g}\\ +\text{(boundary condition)}_1\\ +\text{(boundary condition)}_2. \end{cases}$$

Here $\mathcal{A}_d=\mathcal{A}_d(x_1,x_2)$ and $\mathcal{B}_d=\mathcal{B}_d(x_1,x_2)$ are $n_c\times n_c$ matrices. $\mathbf{u}$ is an n_c-vector containing the solution functions, and (boundary condition)$_d$ denotes a set of boundary conditions in the x_d-direction. Our main interest is to develop efficient solution methods for first-order systems of PDEs, exploiting centered finite-difference approximations. Then artificial viscosity must usually be added to damp high-frequency oscillations. Hence, it is important to take second-order terms into account. The methods presented are applicable both to problems with viscosity in the PDE and to first-order PDEs where artificial viscosity is added only in the difference approximation.

The PDE (1) is discretized using a five-point stencil on a uniform $m_1\times m_2$ grid. Note that nonuniform grids may be accounted for by solving a transformed PDE on a grid with constant stepsize. The methods presented here can be generalized to a $(2q_1+1)\times(2q_2+1)$ rectangular stencil. Thus, it is possible to treat higher-order approximations and mixed second-order terms. Furthermore, nonrectangular grids may be handled by using a domain decomposition technique. These generalizations will be described and analyzed in future work.

We use the notation

$$\operatorname{diag}_{j,m}(\beta_j)=\begin{pmatrix}\beta_1&&\\&\ddots&\\&&\beta_m\end{pmatrix},\quad \operatorname{ptrid}_{j,m}(\alpha_j,\beta_j,\gamma_j)=\begin{pmatrix}\beta_1&\gamma_1&&\alpha_1\\ \alpha_2&\ddots&\ddots&\\ &\ddots&\ddots&\gamma_{m-1}\\ \gamma_m&&\alpha_m&\beta_m\end{pmatrix}.$$

*Received by the editors June 4, 1993; accepted for publication (in revised form) December 26, 1994. This work was supported by the Swedish National Board for Industrial and Technical Development (NUTEK).

†Department of Scientific Computing, Uppsala University, Box 120, S-751 04 Uppsala, Sweden (sverker@tdb.uu.se and kurt@tdb.uu.se).

At the boundaries we impose the restriction that the difference stencil may not be wider than the five-point stencil used in the interior. Hence, e.g., periodic, Dirichlet, and Neumann boundary conditions are allowed. Also numerical outflow boundary conditions may be used in the difference approximation. Note that for hyperbolic PDEs, outflow boundary conditions are always required for the characteristic variables corresponding to outgoing characteristics. Outflow boundary conditions may also be needed for incompletely parabolic problems, e.g., the Navier–Stokes equations.

An approximate solution u of (1) is obtained by solving

$$M^{-1}Bu = M^{-1}g. \tag{2}$$

Here B is the coefficient matrix arising from the discretization of (1), and M is a preconditioner matrix. If we order the $n = n_c m_1 m_2$ unknowns in u first by equation in the system of PDEs and then lexicographically in the domain, B has the following structure:

$$B = \operatorname{ptrid}_{k,m_2}(B_{k,-1}, B_{k,0}, B_{k,1}),$$

where $B_{k,-1} = \operatorname{diag}_{j,m_1}(\nu_{j,k,-2})$ and $B_{k,1} = \operatorname{diag}_{j,m_1}(\nu_{j,k,2})$. The matrix $B_{k,0}$ is given by $B_{k,0} = \operatorname{ptrid}_{j,m_1}(\nu_{j,k,-1}, \nu_{j,k,0}, \nu_{j,k,1})$. Here $\nu_{j,k,r}$, $r = -2, \dots, 2$, are $n_c \times n_c$ matrices. The matrices B or $B_{k,0}$ are periodic tridiagonal only if the PDE is subject to periodic boundary conditions in the x_2- or x_1-direction.

The matrix B is very sparse and highly structured. The number of nonzero elements is only $\sim 5n_c n$. For the PDEs we are interested in, B may be highly nondiagonally dominant. If we impose the restriction that B is nonperiodic tridiagonal, then B is a band matrix with semibandwidth $n_c m_1$. Using band Gaussian elimination with pivoting, the storage and arithmetic requirements grow rapidly with increasing problem size. Instead it is natural to study iterative methods, since multiplying a vector by B requires only $\sim 10n_c n$ arithmetic operations (a.o.). However, iterating on the matrix B, the number of iterations required will generally grow when the problem size is increased. A preconditioner M must be employed to attain acceptable convergence rates for large problems. The preconditioner solve has to be performed by an efficient algorithm. The preconditioner is introduced to obtain acceptable execution time and not to minimize the number of iterations. The preconditioner must also exploit the sparsity of B, so that the storage requirement does not become prohibitive. Many approaches for constructing preconditioners have been considered, e.g., incomplete LU factorization [VdVo82] and polynomial preconditioners [Ashby87]. We use a PDE approach; i.e., M is defined by the system of equations

$$Mu = g, \tag{3}$$

arising from a PDE closely related to (1). The objective is to find simplified PDEs such that it is possible to solve (3) efficiently, and that the number of iterations required to solve (2) is independent of problem size. Apart from yielding preconditioners, efficient solution methods for (3) may also be directly applicable to some PDEs of practical interest.

For constant coefficient elliptic PDEs subject to periodic or Dirichlet boundary conditions, M has a circulant or Toeplitz structure. Then (3) can be solved in $\mathcal{O}(n \log_2 n)$ complexity using a fast Poisson solver based on fast trigonometric transforms [VLoan92]. For constant coefficient hyperbolic or incompletely parabolic PDEs subject to periodic boundary conditions, M again has a circulant structure allowing a solver based on fast Fourier transforms. However, for PDE problems subject to Dirichlet boundary conditions, this is not the case. As previously mentioned, numerical outflow boundary conditions must be supplied in the difference approximation. Thus, the Toeplitz structure of M will be disturbed at the entries corresponding to the boundaries. The purpose of this paper is to study solvers for problems of this type. We use the fast transform framework but extend it to allow the disturbances arising from the outflow boundary conditions.

In previous papers we have described *semicirculant* (SC) solvers. Here, we use a specific SC solver described by $M_{(SC)}$ corresponding to a discretization of a PDE related to (1) but where two approximations are introduced. The coefficients are approximated by constants in the x_1-direction, and the boundary conditions in the same space direction are changed to periodic. In [HoOtto92], [Otto97], and [HoOtto94], SC solvers are used as preconditioners for problems of type (2), where B corresponds to first-order time-dependent and time-independent PDEs subject to Dirichlet boundary conditions. Empirically, the number of iterations required is found to be independent of problem size for both problem types. In [Otto97] and [HoOtto94], it is also proved that both the eigenvalues and the eigenvectors of $M_{(SC)}^{-1}B$ are "improved" compared with those of B. The SC solver is the basis for the new solution methods presented in this paper, and it is briefly described in §2.

In §3 we present the boundary corrected semicirculant (BCSC) framework. Also the matrix $M_{(BCSC)}$ corresponds to a PDE closely related to (1). We approximate the coefficients in the x_1-direction by constants, whereas the original boundary conditions are retained. The basis for the BCSC framework is a factorization for $M_{(BCSC)}^{-1}$, given by the Sherman-Morrison-Woodbury formula. This factorization could be used in different ways. We first derive an efficient direct solver for (3). We also use the factorization to construct an iteration for (3), which resembles the Schur complement matrix method used in domain decomposition [KeyGr87]. In §6 we prove that the eigendecomposition of the iteration matrix is closely related to that of $M_{(SC)}^{-1}M_{(BCSC)}$. This makes it possible to exploit theoretical results for model problems derived in [Otto97] and [HoOtto94].

In §4 we summarize the arithmetic complexities and storage requirements for the solvers, and in §5 two model problems are presented. Sections 7 and 8 contain numerical results for these PDEs. Finally, in §9 the BCSC solver is employed to solve the Navier-Stokes equations.

2. The SC solver. In this section a brief review of the SC framework is given. For a more complete discussion on SC preconditioners and some numerical results, see [HoOtto90], [HoOtto92], [Otto97], and [HoOtto94].

Define the circulant tridiagonal matrix $\mathrm{ctrid}_m(\alpha, \beta, \gamma)$ by

$$\mathrm{ctrid}_m(\alpha,\beta,\gamma) = \begin{pmatrix} \beta & \gamma & & \alpha \\ \alpha & \ddots & \ddots & \\ & \ddots & \ddots & \gamma \\ \gamma & & \alpha & \beta \end{pmatrix}.$$

The matrix $M_{(SC)}$ is defined by

$$M_{(SC)} = \mathrm{ptrid}_{k,m_2}(M_{k,-1}, M_{k,0}^{(SC)}, M_{k,1}), \tag{4}$$

where

$$M_{k,-1} = I_{m_1} \otimes \rho_{k,-2}, \quad M_{k,0}^{(SC)} = \mathrm{ctrid}_{m_1}(\rho_{k,-1}, \rho_{k,0}, \rho_{k,1}), \quad M_{k,1} = I_{m_1} \otimes \rho_{k,2}.$$

The matrix $M_{(SC)}$ is as sparse as B and completely described by the $5m_2$ $n_c \times n_c$ matrices $\rho_{k,r}$. Several choices for these parameters are possible [HoOtto92], but when using the PDE approach a natural choice is

$$\rho_{k,r} = \frac{1}{m_1}\sum_{j=1}^{m_1} \tilde{v}_{j,k,r}, \qquad k = 1, \ldots, m_2, \qquad r = -2, \ldots, 2,$$

where $\tilde{v}_{j,k,r}$ is a set of matrix entries forming the matrix $\tilde{B}$, corresponding to the difference approximation of (1), but subject to periodic boundary conditions in the x_1-direction. Thus, $M_{(SC)}$ is a discretization of the PDE

$$(5)\qquad \begin{cases} \left(\dfrac{\partial \mathbf{u}}{\partial t}+\right) \tilde{\mathcal{A}}_1 \dfrac{\partial \mathbf{u}}{\partial x_1} + \tilde{\mathcal{B}}_1 \dfrac{\partial^2 \mathbf{u}}{\partial x_1^2} + \tilde{\mathcal{A}}_2 \dfrac{\partial \mathbf{u}}{\partial x_2} + \tilde{\mathcal{B}}_2 \dfrac{\partial^2 \mathbf{u}}{\partial x_2^2} = \mathbf{g} \\ + \text{periodic boundary conditions in the } x_1\text{-direction} \\ + (\text{boundary condition})_2. \end{cases}$$

Here $\tilde{\mathcal{A}}_d = \tilde{\mathcal{A}}_d(x_2)$ and $\tilde{\mathcal{B}}_d = \tilde{\mathcal{B}}_d(x_2)$ are defined as averages over x_1 of $\mathcal{A}_d(x_1, x_2)$ and $\mathcal{B}_d(x_1, x_2)$.

The reason for introducing circulant matrices in $M_{(SC)}$ is that such matrices are diagonalized by using discrete Fourier transforms, which are performed by the fast Fourier transform (FFT) algorithm. For each new SC preconditioner, i.e., each new set of parameters $\rho_{k,r}$, $k = 1, \ldots, m_2$, $r = -2, \ldots, 2$, a factorization phase requiring $\mathcal{O}(n)$ a.o. is performed once. The arithmetic complexity for each SC solve is $\mathcal{O}(n \log_2 m_1)$, and the memory requirement is $\mathcal{O}(n)$.

3. The BCSC solver. We now study solution methods for (3), arising from the discretization of the PDE

$$(6)\qquad \begin{cases} \left(\dfrac{\partial \mathbf{u}}{\partial t}+\right) \tilde{\mathcal{A}}_1 \dfrac{\partial \mathbf{u}}{\partial x_1} + \tilde{\mathcal{B}}_1 \dfrac{\partial^2 \mathbf{u}}{\partial x_1^2} + \tilde{\mathcal{A}}_2 \dfrac{\partial \mathbf{u}}{\partial x_2} + \tilde{\mathcal{B}}_2 \dfrac{\partial^2 \mathbf{u}}{\partial x_2^2} = \mathbf{g} \\ + (\text{boundary condition})_1 \\ + (\text{boundary condition})_2. \end{cases}$$

The direct BCSC solver is derived, but also an iterative solver is described. The matrix $M_{(BCSC)}$ is defined by

$$(7)\qquad M_{(BCSC)} = \operatorname{ptrid}_{k,m_2}(M_{k,-1}, M_{k,0}^{(BCSC)}, M_{k,1}),$$

where $M_{k,-1}$ and $M_{k,1}$ are defined in (4), and $M_{k,0}^{(BCSC)}$ is given by

$$M_{k,0}^{(BCSC)} = \begin{pmatrix} \rho_{k,0}+\xi_{k,0} & \rho_{k,1}+\xi_{k,1} & & & \\ \rho_{k,-1} & \rho_{k,0} & \rho_{k,1} & & \\ & \ddots & \ddots & \ddots & \\ & & \rho_{k,-1} & \rho_{k,0} & \rho_{k,1} \\ & & & \rho_{k,-1}+\zeta_{k,-1} & \rho_{k,0}+\zeta_{k,0} \end{pmatrix}.$$

The disturbances of the Toeplitz structure arising from (boundary condition)$_1$ are now accounted for.

The solution methods for (3) described below are based on SC solves. We first note that $M_{(BCSC)} - M_{(SC)}$ is very sparse, and

$$(8)\qquad E \equiv M_{(BCSC)} - M_{(SC)} = SV^T,$$

where $S = I_{m_2} \otimes s$ and $V = \operatorname{diag}_{k,m_2}(v_k)$. Here s and v_k are $n_c m_1 \times 2n_c$ matrices given by

$$s^T = \begin{pmatrix} I_{n_c} & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & I_{n_c} \end{pmatrix}$$

and

$$v_k^T = \begin{pmatrix} \xi_{k,0} & \xi_{k,1} & 0 & \cdots & 0 & 0 & -\rho_{k,-1} \\ -\rho_{k,1} & 0 & 0 & \cdots & 0 & \zeta_{k,-1} & \zeta_{k,0} \end{pmatrix}.$$

Exploiting the Sherman-Morrison-Woodbury formula, a factorization for $M_{(BCSC)}^{-1}$ is obtained:

$$M_{(BCSC)}^{-1} = (M_{(SC)} + SV^T)^{-1} = (I - M_{(SC)}^{-1} SP^{-1}V^T)M_{(SC)}^{-1}, \tag{9}$$

where

$$P = I + V^T M_{(SC)}^{-1} S. \tag{10}$$

This factorization can be used in several ways. We first describe the BCSC solver. The two multiplications by $M_{(SC)}^{-1}$ in (9) can be performed as SC solves. Exploiting the sparsity of V, it is relatively cheap to form the $2n_c m_2 \times 2n_c m_2$ matrix P. For simplicity, we only show the computation of P for the case $n_c = 1$, but the generalization to blocks is straightforward. The algorithm is given in a MATLAB style.

```
ALGORITHM FOR COMPUTING $P$
for $\ell = 1$ to $2m_2$ do
    $z = M_{(SC)}^{-1} S(:, \ell)$
    for $k = 1$ to $m_2$ do
        $k_1 = (k-1)m_1$
        $P(2k-1, \ell) = \xi_{k,0} z(k_1+1) + \xi_{k,1} z(k_1+2) - \rho_{k,-1} z(k_1+m_1)$
        $P(2k, \ell) = -\rho_{k,1} z(k_1+1) + \zeta_{k,-1} z(k_1+m_1-1) + \zeta_{k,0} z(k_1+m_1)$
    endfor
endfor
$P = I + P$
```

Using SC solves, $\mathcal{O}(m_2 n \log_2 m_1)$ a.o. are required to form P. The factorization $P = LU$ can be computed in $\mathcal{O}(m_2^3)$ complexity. In §6 we prove that P^{-1} exists for some first-order model problems. Now $u = M_{(BCSC)}^{-1} g$ is computed according to the following method.

```
BCSC SOLVE
        $u \leftarrow M_{(SC)}^{-1} g$
        $z \leftarrow SU^{-1}L^{-1}V^T u$
        $z \leftarrow M_{(SC)}^{-1} z$
        $u \leftarrow u - z$
```

Here $\mathcal{O}(n \log_2 m_1)$ a.o. are required. Note that, excluding storage required for the SC solver, only storage for the factorization of P and an auxiliary vector of length n is required. Hence, the storage requirement for the BCSC solver is $\mathcal{O}(n)$.

If it is considered too expensive to form P, P^{-1} may be approximated by some matrix H which is cheaper to form. Then the matrix $M_H^{-1} \equiv (I - M_{(SC)}^{-1} SHV^T)M_{(SC)}^{-1}$ could be used as a preconditioner in an iteration for (3). It is a trivial matter to show that $\operatorname{rank}(M_H^{-1} - M_{(BCSC)}^{-1}) \leq 2n_c m_2$, implying that $M_H^{-1} M_{(BCSC)}$ has at most $2n_c m_2$ eigenvalues separated from 1. Note that the choice $H = 0$ yields $M_H = M_{(SC)}$.

It is also possible to utilize the factorization (9) to derive another iteration for (3), which is in some sense related to the Schur complement matrix method used in domain decomposition. When computing $u = M_{(BCSC)}^{-1} g$, operations of the type $x = P^{-1} y$ must be performed. If we solve $Px = y$ using a CG-like iterative method, P need not be formed, since only

TABLE 1
Arithmetic complexities for factorization.

Solver	Arithmetic operations
bandGE SC BCSC	between $\sim 2n_c^2 m_1^2 n$ and $\sim 4n_c^2 m_1^2 n$ $\sim (32n_c^2 - 21n_c + 3)n$ $\sim \big((10\log_2 m_1 + 48n_c + 4)n_c m_2 + 32n_c^2 - 21n_c + 3\big)n$ $+ 24n_c^3 m_2^2 + \frac{16}{3}n_c^3 m_2^3 - 2n_c^2 m_2^2$

TABLE 2
Arithmetic complexities for substitutions.

Solver	Arithmetic operations
bandGE SC BCSC	$\sim 6n_c m_1 n$ $\sim (5\log_2 m_1 + 24n_c + 2)n$ $\sim (10\log_2 m_1 + 48n_c + 5)n + 8n_c^2 m_2^2$

TABLE 3
Storage requirements.

Solver	Memory positions
bandGE SC BCSC	$\sim 3n_c m_1 n$ $\sim 6n_c n$ $\sim (6n_c + 1)n + 4n_c^2 m_2^2$

multiplications by $P = I + V^T M_{(SC)}^{-1} S$ are required. Due to the sparsity of S and V, $y = Px$ can be performed using basically one SC solve. Multiplying a vector by S requires no arithmetic, and the complexity for a multiplication by V^T is only $\mathcal{O}(m_2)$. Note that x and y are $2n_c m_2$-vectors. We iterate on the boundaries in the x_1-direction, whereas an exact solver is used in the interior of the domain. This iteration could be used for solving (3) or as an inner iteration for the preconditioner when solving (2).

4. Summary of arithmetic and storage requirements. In [Holm92] the implementation of the SC solver is considered, and savings in arithmetic complexity, generalizations, and parallelization are discussed. In Tables 1 and 2, we summarize the arithmetic complexities for the Gaussian elimination (bandGE), SC, and BCSC solvers. In Table 3 we give the corresponding storage requirements for the implementation used here.

In §§7 and 8, we use the restarted GMRES method denoted by GMRES(ℓ) [SaadSch86]. The average number of arithmetic operations required for one iteration is $(2\ell + 10)n$, excluding the work required for the matrix-vector multiplication and preconditioner solve. The storage requirement is $(\ell + 2)n$. We use the initial guess $u_0 = 0$ and the following stopping criterion:

$$\frac{\|M^{-1}(g - Bu_i)\|_2}{\|M^{-1}g\|_2} \le 10^{-6}.$$

5. Model problems. As representative model problems, we use a scalar hyperbolic PDE subject to Dirichlet boundary conditions in a time-dependent and a time-independent setting. The time-dependent problem is given by

$$\frac{\partial \mathbf{u}}{\partial t} + \sigma_1(x_1)\frac{\partial \mathbf{u}}{\partial x_1} + \sigma_2(x_2)\frac{\partial \mathbf{u}}{\partial x_2} = \mathbf{g}, \tag{11}$$

on the unit square for $t > 0$. We assume that $\sigma_d(x_d) > 0$. The PDE (11) is well posed if $\mathbf{u}(0, x_2, t)$, $\mathbf{u}(x_1, 0, t)$, and $\mathbf{u}(x_1, x_2, 0)$ are supplied. Note that the solution must not be prescribed at the outflow boundaries.

The time discretization is performed using the second-order accurate trapezoidal rule with time-step Δt. The spatial discretization is performed on a uniform grid with $(m_1 + 1) \times (m_2 + 1)$ gridpoints. Hence, the space-step in the x_d-direction is given by $h_d \equiv 1/m_d$. The spatial derivatives in (11) are approximated using second-order accurate centered differences in the interior of the domain. For the numerical boundary conditions required at the outflow boundaries, we use one-sided differences. Let $\kappa_{j,d} \equiv \sigma_d(jh_d)\Delta t/h_d$, $j = 1, \ldots, m_d$. If $\sigma_d(x_d) = \sigma_d$, then $\kappa_{j,d} = \kappa_d \equiv \sigma_d \Delta t/h_d$. Introducing the discretizations in (11) yields the following system of equations for the solution at time level $N + 1$:

$$Bu^{N+1} \equiv (I_{m_2} \otimes B_1 + B_2 \otimes I_{m_1})u^{N+1} = g. \tag{12}$$

Here g contains known quantities and

$$B_1 = \begin{pmatrix} 4 & \kappa_{1,1} & & & \\ -\kappa_{2,1} & 4 & \kappa_{2,1} & & \\ & \ddots & \ddots & \ddots & \\ & & -\kappa_{m_1-1,1} & 4 & \kappa_{m_1-1,1} \\ & & & -2\kappa_{m_1,1} & 4 + 2\kappa_{m_1,1} \end{pmatrix},$$

$$B_2 = \begin{pmatrix} 0 & \kappa_{1,2} & & & \\ -\kappa_{2,2} & 0 & \kappa_{2,2} & & \\ & \ddots & \ddots & \ddots & \\ & & -\kappa_{m_2-1,2} & 0 & \kappa_{m_2-1,2} \\ & & & -2\kappa_{m_2,2} & 2\kappa_{m_2,2} \end{pmatrix}.$$

A more detailed description is given in [HoOtto90]. We also consider the time-independent version of (11):

$$\sigma_1(x_1)\frac{\partial \mathbf{u}}{\partial x_1} + \sigma_2(x_2)\frac{\partial \mathbf{u}}{\partial x_2} = \mathbf{g}. \tag{13}$$

The boundary conditions are of the same form as for the time-dependent problem. Also the discretization in space is performed on the same type of grid, and the derivatives in the PDE are approximated using centered differences in the interior of the domain. However, we add a weak artificial viscosity in the x_1-direction. The difference operator in the interior is given by

$$\sigma_1(x_1)\left(D_{0,x_1} - \frac{\gamma}{2}h_1^{2-\alpha}D_{+,x_1}D_{-,x_1}\right) + \sigma_2(x_2)D_{0,x_2},$$

where $D_{+,x_1}u_{j,k} \equiv h_1^{-1}(u_{j+1,k} - u_{j,k})$, $D_{-,x_1}u_{j,k} \equiv D_{+,x_1}u_{j-1,k}$, and finally $D_{0,x_1} \equiv \frac{1}{2}(D_{+,x_1} + D_{-,x_1})$. Here $u_{j,k}$ is the approximation of $\mathbf{u}$ at the gridpoint (jh_1, kh_2). The constants α and γ are chosen so that $\alpha \in (0, 1)$ and $\gamma \in (0, 1)$. The order of accuracy is $2 - \alpha$ in the x_1-direction and 2 in the x_2-direction. For the numerical boundary conditions we again use one-sided differences. A more detailed description of a model problem of this type and a discussion on the choice of artificial viscosity is given in [HoOtto94]. Define $\delta_j \equiv \sigma_1(jh_1)\gamma h_1^{-\alpha}$ and $\hat{\kappa}_{j,d} \equiv \sigma_d(jh_d)h_d^{-1}$. Also define $\tau_{j,-1} \equiv -\hat{\kappa}_{j,1} - \delta_j$ and $\tau_{j,1} \equiv \hat{\kappa}_{j,1} - \delta_j$. If $\sigma_d(x_d) = \sigma_d$, then

$$\begin{aligned} \delta_j &= \delta \equiv \sigma_1\gamma h_1^{-\alpha}, \\ \hat{\kappa}_{j,d} &= \hat{\kappa}_d \equiv \sigma_d h_d^{-1}, \\ \tau_{j,-1} &= \tau_{-1} \equiv -\hat{\kappa}_1 - \delta, \\ \tau_{j,1} &= \tau_1 \equiv \hat{\kappa}_1 - \delta. \end{aligned}$$

By introducing the discretization in (13), the following system of equations is derived:

$$Bu \equiv (I_{m_2} \otimes \hat{B}_1 + \hat{B}_2 \otimes I_{m_1})u = g. \tag{14}$$

Again g contains known quantities and

$$\hat{B}_1 = \begin{pmatrix} 2\delta_1 & \tau_{1,1} & & & \\ \tau_{2,-1} & 2\delta_2 & \tau_{2,1} & & \\ & \ddots & \ddots & \ddots & \\ & & \tau_{m_1-1,-1} & 2\delta_{m_1-1} & \tau_{m_1-1,1} \\ & & & -2\hat{\kappa}_{m_1,1} & 2\hat{\kappa}_{m_1,1} \end{pmatrix},$$

$$\hat{B}_2 = \begin{pmatrix} 0 & \hat{\kappa}_{1,2} & & & \\ -\hat{\kappa}_{2,2} & 0 & \hat{\kappa}_{2,2} & & \\ & \ddots & \ddots & \ddots & \\ & & -\hat{\kappa}_{m_2-1,2} & 0 & \hat{\kappa}_{m_2-1,2} \\ & & & -2\hat{\kappa}_{m_2,2} & 2\hat{\kappa}_{m_2,2} \end{pmatrix}.$$

Note that the coefficient matrices are denoted by B for both model problems.

6. Analysis of the BCSC framework. In this section we derive formulas for the eigenvalues and eigenvectors of the matrix P defined in (10). The eigendecomposition of P is given in terms of the eigenvalues and eigenvectors of $M_{(SC)}^{-1}E$, where E is defined in (8). For first-order model problems, the eigendecomposition of $M_{(SC)}^{-1}E$ has been studied in [Otto97] and [HoOtto94]. In Theorem 2 we use this information to prove that P^{-1} exists.

THEOREM 1. *Let the eigendecomposition of $M_{(SC)}^{-1}E$ be given by*

$$(M_{(SC)}^{-1}E)W_{(SC)} = W_{(SC)}\Lambda_{(SC)},$$

where

$$\Lambda_{(SC)} = \mathrm{diag}_{k,m_2}(\Lambda_k^{(SC)}), \quad \Lambda_k^{(SC)} = \mathrm{diag}_{j,m_1}(\lambda_{j,k}^{(SC)}).$$

Then the eigenvectors of P are given by $W_P = V^T W_{(SC)} S$. Also, the $2m_2$ eigenvalues $\lambda_{(1,2),k}^{(P)}$ are $1 + \lambda_{1,k}^{(SC)}$ and $1 + \lambda_{m_1,k}^{(SC)}$, $k = 1, \dots, m_2$.

Proof. Recall that $P = I + V^T M_{(SC)}^{-1} S$. We have

$$\begin{aligned} P(V^T W_{(SC)} S) &= V^T W_{(SC)} S + V^T M_{(SC)}^{-1} S V^T W_{(SC)} S \\ &= V^T W_{(SC)} S + V^T (M_{(SC)}^{-1}E) W_{(SC)} S = V^T W_{(SC)} (I + \Lambda_{(SC)}) S. \end{aligned}$$

Also, exploiting the structure of S we obtain

$$\begin{aligned} (I + \Lambda_{(SC)})S &= S + \mathrm{diag}_{k,m_2}(\Lambda_k^{(SC)})(I_{m_2} \otimes s) = S + \mathrm{diag}_{k,m_2}(\Lambda_k^{(SC)} s) \\ &= \mathrm{diag}_{k,m_2}(s\Lambda_k^{(P)}) = (I_{m_2} \otimes s)\mathrm{diag}_{k,m_2}(\Lambda_k^{(P)}) = S\Lambda_P, \end{aligned}$$

where

$$\Lambda_P = \mathrm{diag}_{k,m_2}(\Lambda_k^{(P)}),$$

$$\Lambda_k^{(P)} = \begin{pmatrix} \lambda_{1,k}^{(P)} & \\ & \lambda_{2,k}^{(P)} \end{pmatrix} = \begin{pmatrix} 1 + \lambda_{1,k}^{(SC)} & \\ & 1 + \lambda_{m_1,k.}^{(SC)} \end{pmatrix}.$$

Thus,

$$P(V^T W_{(SC)} S) = (V^T W_{(SC)} S)\Lambda_P.$$

Hence, $W_P = V^T W_{(SC)} S$ is the eigenvector matrix to P, and $\lambda_{1,k}^{(P)} = 1+\lambda_{1,k}^{(SC)}, \lambda_{2,k}^{(P)} = 1+\lambda_{m_1,k}^{(SC)}$ are eigenvalues of P. □

If $\sigma_1(x_1) = \sigma_1$, the model problems in §5 satisfy $B = M_{(BCSC)}$. According to (8) this implies

$$M_{(SC)}^{-1} B = M_{(SC)}^{-1}(M_{(SC)} + E) = I + M_{(SC)}^{-1} E,$$

and the eigenvalues of P are trivially extracted. For the time-dependent model problem, v_k is given by

$$v_k^T = \begin{pmatrix} 0 & 0 & \cdots & 0 & 0 & \kappa_1 \\ -\kappa_1 & 0 & \cdots & 0 & -\kappa_1 & 2\kappa_1 \end{pmatrix},$$

whereas for the time-independent problem

$$v_k^T = \begin{pmatrix} 0 & 0 & \cdots & 0 & 0 & -\tau_{-1} \\ -\tau_1 & 0 & \cdots & 0 & -\tau_1 & 2\tau_1. \end{pmatrix}$$

Now assume that $\sigma_d(x_d) = \sigma_d = 1$. In [HoOtto94] formulas relating the eigendecomposition of $M_{(SC)}^{-1} E$ to the eigendecomposition of $\hat{\kappa}_2^{-1}\hat{B}_2$ are given. In [Otto97] the corresponding formulas for the time-dependent model problem are given. The two problem settings are closely related. In the discussion below, we use the time-independent problem and the notation in [HoOtto94]. By performing the relevant substitutions, the corresponding results for the time-dependent problem follow.

In [HoOtto94] it is proved that $M_{(SC)}^{-1} E$ has at most $2m_2$ nonzero eigenvalues, denoted by $\lambda_{1,k}^{(SC)}$ and $\lambda_{m_1,k}^{(SC)}$, $k = 1, \ldots, m_2$. Furthermore, it is proved that $M_{(SC)}^{-1} E$ is diagonalizable, and asymptotic formulas for the eigenvalues of $M_{(SC)}^{-1} E$ are derived. Under certain conditions it is established that, for large values of m_1 and m_2, $\lambda_{1,k}^{(SC)}$ and $\lambda_{m_1,k}^{(SC)}$ approach two finite curve segments $\lambda_{1,\infty}^{(SC)}(\theta)$, where $|\theta| \le \varphi < 1$. The curve segments are identical for the time-dependent and time-independent model problems. For the time-independent problem we have Theorem 2.

THEOREM 2. *If $\sigma_d(x_d) = \sigma_d = 1$, m_1 and m_2 are sufficiently large, and*

$$\frac{m_2(1 + 2m_2^{-3/2})}{m_1(1 - \gamma m_1^{\alpha-1})} \le \varphi, \quad \textit{for some constant } \varphi < 1,$$

then P^{-1} exists.

Proof. Under the given assumptions, Corollary 1 in [HoOtto94] yields

$$\frac{15}{28} < |1 + \lambda_{1,\infty}^{(SC)}(\theta)| < 1 + \frac{\sqrt{5}+\sqrt{7}}{4\sqrt{1-\varphi^2}}, \quad -\varphi \le \theta \le \varphi.$$

Hence, for sufficiently large values of m_1 and m_2, we have $|\lambda_{1,k}^{(P)}| = |1 + \lambda_{1,k}^{(SC)}| > 0$ and $|\lambda_{2,k}^{(P)}| = |1 + \lambda_{m_1,k}^{(SC)}| > 0, k = 1, \ldots, m_2$. □

When $m_1, m_2 \to \infty$, the condition on the grid in Theorem 2 can be reduced to $m_2/m_1 = \varphi < 1$. Normally, this is also the condition used for practical computations. Note that Theorem 2 only gives sufficient conditions for the existence of P^{-1}.

TABLE 4
The condition number of the eigenvector matrices.

	$m = 8$	$m = 12$	$m = 16$	$m = 24$	$m = 32$
$\mathrm{cond}_2(W_P)$	12.8	21.5	32.4	58.3	90.3
$\mathrm{cond}_2(W_{(SC)})$	$1.29 \cdot 10^3$	$1.05 \cdot 10^4$	$8.78 \cdot 10^4$	$3.64 \cdot 10^8$	not computable

Assume that a system of n equations

$$Ax = b$$

is solved using a *minimal residual* iteration [FrGoNa92], e.g., the GMRES method. Then we have

$$\|r_i\|_2 = \min_{p_i \in \mathcal{P}_i,\, p_i(0)=1} \|p_i(A) r_0\|_2,$$

where $r_i = Ax_i - b$ and x_i is the approximation of x computed in iteration i. If A is diagonalizable, we obtain [FrGoNa92]

$$\frac{\|r_i\|_2}{\|r_0\|_2} \le \mathrm{cond}_2(W_A) \cdot \min_{p_i \in \mathcal{P}_i,\, p_i(0)=1} \max_{1 \le \ell \le n} |p_i(\lambda_\ell)| \equiv \mathrm{cond}_2(W_A) \cdot \varepsilon_i(A). \tag{15}$$

Here W_A is the eigenvector matrix and λ_ℓ the eigenvalues of A. $\mathcal{P}_i$ is the set of all polynomials of maximal degree i. The *asymptotic convergence factor* ϱ_A is defined by

$$\varrho_A \equiv \lim_{i \to \infty} \varepsilon_i(A)^{1/i}.$$

If we can prove that $\varrho_A \le \eta_1 < 1$ and $\mathrm{cond}_2(W_A) \le \eta_2$, where $\eta_1, \eta_2 > 0$ are independent of m_1 and m_2, then asymptotically the number of iterations required does not depend on problem size. Note that (15) may yield a very pessimistic estimate of the convergence rate. It is trivial to construct a problem of arbitrary size where $\mathrm{cond}_2(W_A) = \infty$, and for which a CG-like method converges in two iterations. For non-normal matrices the spectral decomposition framework is not satisfactory, but unfortunately there is no practical theory available that yields a sharp bound on the convergence rate for polynomial iterations.

If the assumptions in Theorem 2 are fulfilled, it is proved [HoOtto94] that, in the limit $m_1, m_2 \to \infty$,

$$\varrho_{M_{(SC)}^{-1}B} < \left(\frac{1 + \frac{40}{225}\sqrt{1-\varphi^2}}{1 + \frac{616}{225}\sqrt{1-\varphi^2}} \right)^{1/2} < 1.$$

Since the eigenvalues of P constitute a subset of the eigenvalues of $M_{(SC)}^{-1}B$, ϱ_P is asymptotically bounded by the same quantity. The eigenvector matrix W_P is more difficult to analyze theoretically. Using the formulas given in [HoOtto94] and a theorem in [HoOtto93] relating W_P to these formulas, such an analysis might be possible. However, we have so far not completed this approach. For the time-independent model problem, W_P turns out to be much better conditioned than $W_{(SC)}$. In Table 4 we show $\mathrm{cond}_2(W_{(SC)})$ and $\mathrm{cond}_2(W_P)$, computed by MATLAB. The parameters for the artificial viscosity are chosen as $\alpha = 0.1$, $\gamma = 0.5$, and $m = m_1 = 2m_2$.

We see that $\mathrm{cond}_2(W_{(SC)})$ is large and grows rapidly when the problem size is increased. The results in Table 4 also show that $\mathrm{cond}_2(W_P)$ grows with increasing problem size, but it is not clear whether this quantity is bounded or not. In Table 5 we show the number of SC

TABLE 5
The number of SC solves required when iterating on P or $M_{(SC)}^{-1}B$.

	$m = 8$	$m = 16$	$m = 32$	$m = 64$	$m = 128$	$m = 256$
P	18	22	18	16	14	14
$M_{(SC)}^{-1}B$	17	22	16	14	14	12

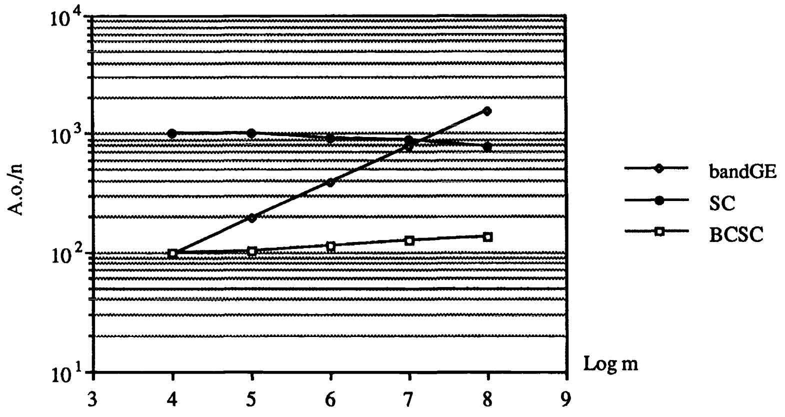


FIG. 1. *Work required for one time-step.*

solves required when the same type of problems as in Table 4 are solved using the GMRES(6) iteration for P or $M_{(SC)}^{-1}B$.

It is clear that there is no significant difference between iterating on P or $M_{(SC)}^{-1}B$. The large values of $\text{cond}_2(W_{(SC)})$ do not affect the number of iterations. As previously remarked, this does not contradict the estimate (15).

7. Results for constant coefficient problems. In this and the next section, we study the model problems described in §5. Assume that $\sigma_d(x_d) = \sigma_d = 1$. Then $M_{(BCSC)}^{-1}B = I$ whereas $M_{(SC)}^{-1}B \neq I$. Hence, the bandGE and BCSC algorithms are used as direct solvers, and the SC solver is used as a preconditioner in the GMRES(6) iteration.

For time-dependent problems, the factorization is performed once prior to the time-marching. If a large number of time-steps are performed, the work required for factorization could be neglected. When the BCSC solver or the SC-preconditioned GMRES(6) iteration is used, the work required to perform one time-step is dominated by SC solves. Therefore, if more than two iterations are required for the GMRES method, the BCSC solver is presumably more efficient. This is verified in Fig. 1, where the number of a.o. per unknown required for one time-step is shown. Problems where $m_1 = 2^p$, $m_2 = \frac{15}{16}2^p$, and $\Delta t = 10 \cdot 2^{-p}$ are studied. Hence, $\kappa_1 = 10$ and $\kappa_2 = 150/16$.

From Fig. 1 it is clear that the BCSC solver is the best choice for all problem sizes studied. However, note that the work required per unknown for the SC-preconditioned GMRES(6) method actually decreases when the problem size is increased. The number of iterations required decreases so fast that the increase in the work required for the SC solve is more than canceled. The number of a.o. required for the bandGE solver becomes unrealistically large when the problem size increases.

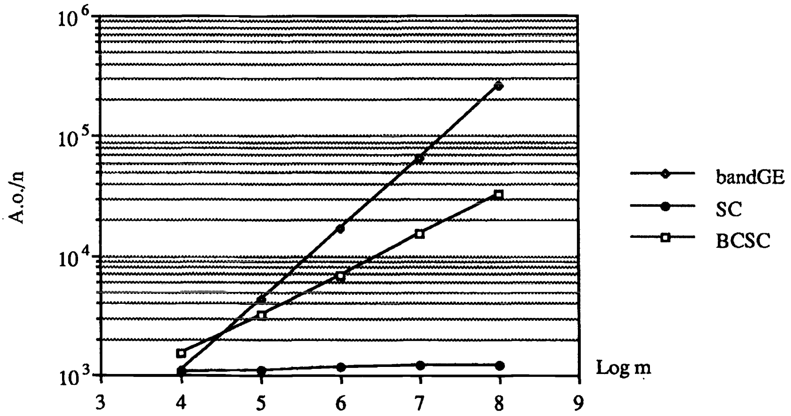


FIG. 2. *Work required for the time-independent problem.*

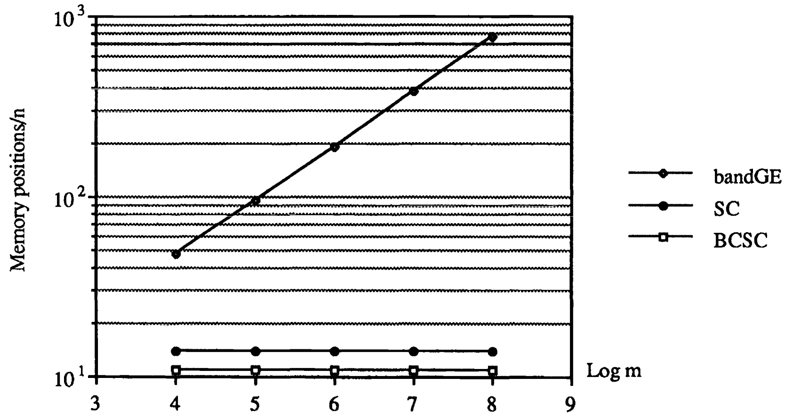


FIG. 3. *Storage requirement.*

When solving time-independent problems, normally only one system of equations has to be solved. Hence, the work required for the factorization must be accounted for. In [HoOtto94] it is found that the number of iterations required for the SC-preconditioned GMRES method is independent of problem size. The factorization for the BCSC solver requires $2n_c m_2$ SC solves. Thus, for large problems the iterative method is presumably better than the BCSC solver. In Fig. 2 the number of a.o. per unknown required for the solution of the time-independent model problem is shown.

Figure 2 clearly demonstrates that the SC-preconditioned GMRES method is the fastest solution method for large problems. Again, the work required does not grow with increasing problem size, and the bandGE solver is not a realistic alternative.

Finally, in Fig. 3 we show the number of memory positions per unknown required for the model problems. Note that the storage requirement is the same for the time-dependent and time-independent problem.

From Fig. 3 we see that, for large problems, the storage requirement for the bandGE solver is not acceptable.

8. Results for variable coefficient problems. In this section we only study the time-dependent model problem. Hence, the work required for factorization is neglected. We assume

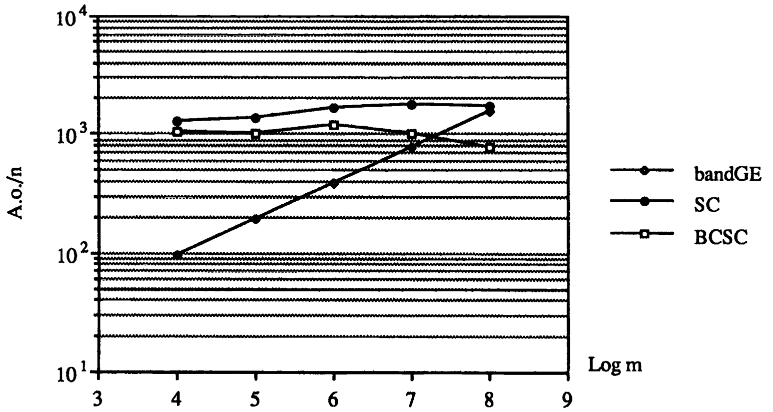


FIG. 4. *Work required for one time-step*, $s = \ln(2)$.

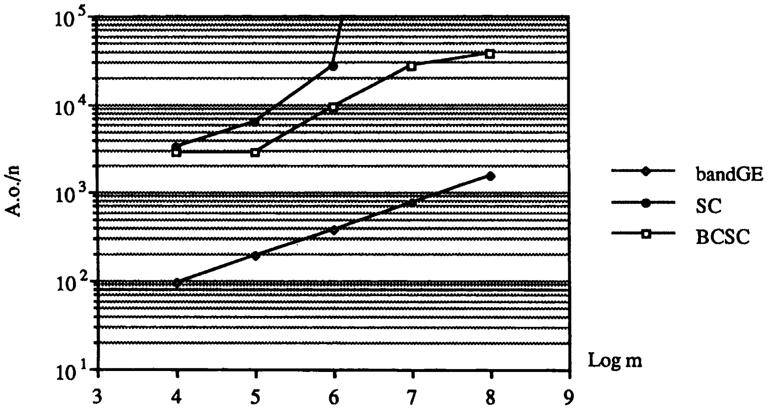


FIG. 5. *Work required for one time-step*, $s = \ln(3)$.

that

$$\sigma_d(x_d) = \sigma(x_d) = \cosh^2\left(s(2x_d - 1)\right)\frac{\tanh(s)}{s}.$$

Here $s > 0$ determines the variation of σ. In the limit $s \to 0$, $\sigma(x_d) = 1$. Coefficients of this form arise from a spatial discretization exploiting a grid adapted to boundary layers; see [HoOtto90]. The variation of $\sigma(x)$ increases rapidly with increasing s. For PDEs with variable coefficients, only the bandGE solver can be used as a direct solution method. Both the SC and BCSC solvers are used as preconditioners in the GMRES(6) iteration.

The BCSC solver takes the boundary conditions in the x_1-direction into account, which is not the case for the SC solver. Hence, $M^{-1}_{(BCSC)}$ is presumably a better approximation of B^{-1} than $M^{-1}_{(SC)}$. In Figs. 4 and 5, the number of a.o. per unknown required for the different solution methods is compared. Problems where $m_1 = 2^p$, $m_2 = \frac{15}{16}2^p$, $\kappa_1 = 10$, and $\kappa_2 = 150/16$ are studied. In Fig. 4 results are shown for $s = \ln(2) \approx 0.69$, whereas in Fig. 5 problems with $s = \ln(3) \approx 1.10$ are examined.

From Fig. 4 it is clear that the BCSC-preconditioned GMRES(6) method requires the smallest number of a.o. for large problems. As anticipated, the BCSC preconditioner performs better than the SC preconditioner. This conclusion is further corroborated by a comparison of numerically computed spectra of $M^{-1}_{(BCSC)}B$ and $M^{-1}_{(SC)}B$ [HoOtto93]. Also note that for this value of s, both the SC and BCSC preconditioners retain the property that the work per unknown does not grow with increasing problem size. For the larger value of s used in Fig. 5, this is no longer true. For both the SC and the BCSC preconditioner, the number of iterations grows when the problem size is increased. The Gaussian elimination algorithm requires the smallest number of a.o. for all problem sizes studied. However, the storage requirement for this algorithm is $\mathcal{O}(m_1)$ times larger than for the iterative methods.

9. An application. We consider the solution of the isentropic Navier-Stokes equations using a semi-implicit time-marching scheme of a type described in [GuSt91]. The specific problem studied here is the calculation of the flow in a driven cavity. The PDE is symmetrized [GuSt91] yielding

$$\frac{\partial \mathbf{u}}{\partial t} + \left(\varepsilon^{-1}\mathcal{P}_0 + \mathcal{P}_1(\mathbf{u})\right)\mathbf{u} = \mathcal{P}_2(\mathbf{u})\mathbf{u}. \tag{16}$$

Here ε is the Mach number and $\mathbf{u}$ is a vector with three components. The first component is a transformed pressure, and the second and third components are the velocities $\mathbf{u}_1$ and $\mathbf{u}_2$ in the x_1- and x_2-directions. For the driven cavity flow both velocity components vanish at all walls, except at the driving boundary $x_2 = 1$, where $\mathbf{u}_1(x_1, 1, t) = 1$ for $t > 0$. The differential operator $\mathcal{P}_2$ contains the nonlinear second-order terms in the Navier-Stokes equations corresponding to viscosity. $\mathcal{P}_1$ is a nonlinear first-order operator, and $\varepsilon^{-1}\mathcal{P}_0$ is a linear first-order operator with constant coefficients. To avoid a strict stability criterion of the form $\Delta t < \mathcal{O}(h\varepsilon)$, an implicit method is preferred for this part of the differential operator. The time-marching method exploits the implicit Euler scheme for $\varepsilon^{-1}\mathcal{P}_0$ and the explicit leap-frog scheme for $\mathcal{P}_1$ and $\mathcal{P}_2$. The discretization of spatial derivatives is performed on a uniform grid with $m \times m$ internal gridpoints and space-step h using centered differences. For smooth solutions and almost incompressible flows, this scheme is second-order time-accurate. Introducing the discretizations, the analytical boundary conditions, and numerical boundary conditions for the transformed pressure, we arrive at a system of equations with $3m^2$ unknowns:

$$Bu = g. \tag{17}$$

Here B is given by

$$B = \begin{pmatrix} B_0 - 2B_1Z & B_1 + B_1Z & & & \\ -B_1 & B_0 & B_1 & & \\ & \ddots & \ddots & \ddots & \\ & & -B_1 & B_0 & B_1 \\ & & & -B_1 - B_1Z & B_0 + 2B_1Z \end{pmatrix},$$

where

$$B_0 = \begin{pmatrix} I_3 - 2\nu_1\zeta & \nu_1 + \nu_1\zeta & & & \\ -\nu_1 & I_3 & \nu_1 & & \\ & \ddots & \ddots & \ddots & \\ & & -\nu_1 & I_3 & \nu_1 \\ & & & -\nu_1 - \nu_1\zeta & I_3 + 2\nu_1\zeta \end{pmatrix},$$

$$B_1 = I_m \otimes \nu_2, \quad Z = I_m \otimes \zeta,$$

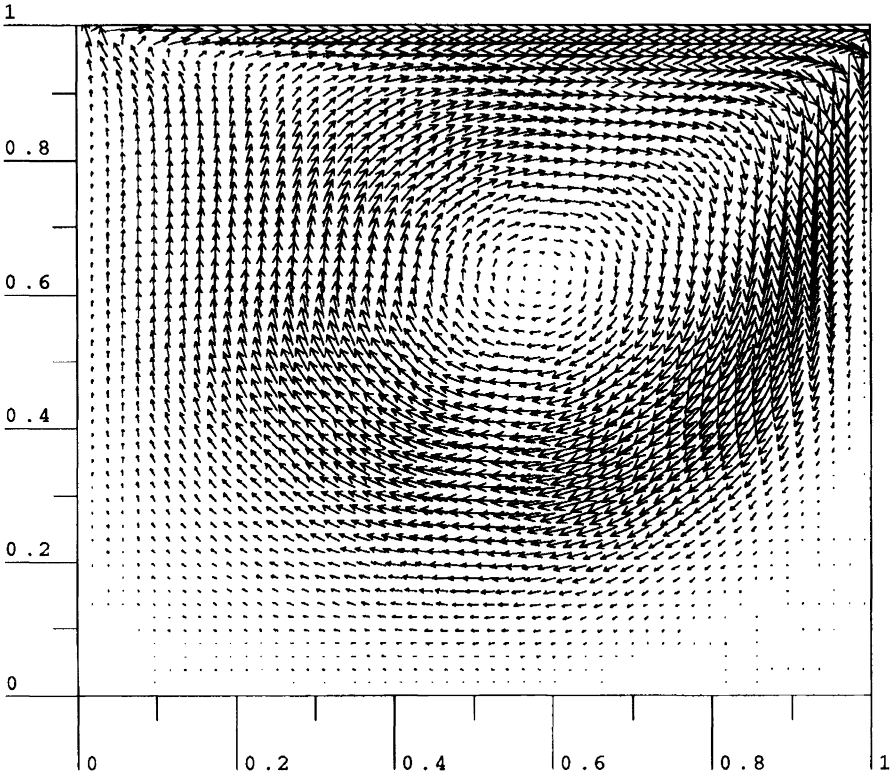


FIG. 6. *The velocity field at* $t = 5$.

and

$$\nu_1 = \begin{pmatrix} 0 & \kappa & 0 \\ \kappa & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \nu_2 = \begin{pmatrix} 0 & 0 & \kappa \\ 0 & 0 & 0 \\ \kappa & 0 & 0 \end{pmatrix}, \quad \zeta = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \kappa = \frac{\Delta t}{\varepsilon h}.$$

In [Stoor90] NAG routines are employed to solve (17). A general sparse matrix factorization routine (F01BRF), exploiting threshold pivoting, is called once prior to the time-marching. In each time-step, F04AXF is invoked to perform the substitutions. Because of the large storage requirement, the largest problem solved in [Stoor90] has only 4800 unknowns.

Below, we show results computed using the BCSC solver, which yields a direct solution method for (17). In Fig. 6 the velocity field at $t = 5$ is shown for a problem with Reynolds number 500. At $t = 0$, the medium was at rest. The Mach number ε is 0.01, which means that the upper boundary of the cavity moves with 1% of the speed of sound. The problem is solved using $m = 256$ and $\Delta t = 0.0005$, yielding $\kappa = 12.85$. Note that the problem studied has 196608 unknowns. The solution is computed using a Sun 670MP. The computer is configured so that $\sim$ 150 Mbyte of virtual memory may be used. Below, problems where $m = 2^p$, $\Delta t = 2^{-(p+2)}$, and $\varepsilon = 0.01$ are studied. In Fig. 7 the number of memory positions per unknown required for the BCSC, NAG, and bandGE solvers is shown. The 150 Mbyte limit is marked as a dashed line.

Figure 7 shows that the BCSC solver allows us to solve a 16 times larger problem than if the NAG solver is used. Note that the quotient between the storage requirement for the bandGE and NAG solvers is $\sim$ 3, independent of problem size. Also note that it is possible to determine

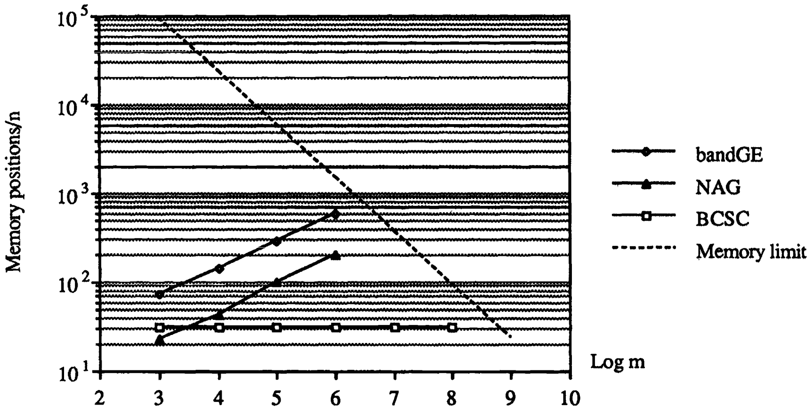


FIG. 7. *Storage requirement.*

TABLE 6
CPU-time (ms/n) required for factorization.

	$m = 16$	$m = 32$	$m = 64$	$m = 128$	$m = 256$
NAG	5.8	87	1060	too large	too large
BCSC	2.7	4.3	10	25	61

TABLE 7
CPU-time (ms/n) required for one time-step.

	$m = 16$	$m = 32$	$m = 64$	$m = 128$	$m = 256$
NAG	0.018	0.035	0.068	too large	too large
BCSC	0.054	0.058	0.063	0.071	0.081

the storage requirement for the BCSC and bandGE solvers at the time of compilation, which is not the case for the NAG solver. When we attempted to use the NAG solver for a problem with $m = 128$, F01BRF spent 38 hours of CPU-time to determine that 396 memory positions per unknown would be required to store the factorization, which is more than the available virtual memory allows!

Tables 6 and 7 show the CPU-time required when the NAG and BCSC solvers are used for the solution of the driven cavity problem. In Table 6 the CPU-time per unknown required for the factorization is shown, and in Table 7 the CPU-time per unknown required for one time-step is displayed. Note that in Table 7 the CPU-time for performing the explicit part of the time-marching is included. The entries "too large" indicate that the corresponding problem is larger than the memory available.

Table 6 clearly shows that the NAG factorization requires much more CPU-time than the BCSC factorization. In Table 7 we see that, for small problems, the NAG substitutions are faster than the BCSC substitutions. However, if it were possible to use the NAG solver for realistic problem sizes, it is clear that the BCSC method would be much faster.

10. Conclusions. Previously, we have described the SC solver characterized by the matrix $M_{(SC)}$. This is a direct solver for systems of PDEs with constant coefficients and periodic boundary conditions in one space direction. For simplicity we now consider $m \times m$ grids, in which case the arithmetic complexity is $\mathcal{O}(m^2 \log_2 m)$. In this paper we have generalized the

SC framework to construct the BCSC solver defined by $M_{(BCSC)}$. Here, the system of PDEs may be subject to boundary conditions from a wide selection, including numerical outflow, Dirichlet, Neumann, and periodic. For quite a large class of PDEs, the SC or BCSC solver is applicable as a direct solution method. Outside the class we have shown how to construct a simplified PDE, for which the solution can be computed using the SC or BCSC solver. The solution of the original problem is then computed with a CG-like iterative method, where the simplified problem acts as a preconditioner.

The basis for the BCSC solver is a factorization of $M_{(BCSC)}^{-1}$, derived by exploiting the Sherman-Morrison-Woodbury formula. To compute the solution, a system of $\mathcal{O}(m)$ equations with coefficient matrix P must be formed and solved. P is a dense matrix containing the boundary corrections. In the BCSC solver, P is formed and LU factorized. The complexity for the first right-hand side is $\mathcal{O}(m^3 \log_2 m)$, whereas only $\mathcal{O}(m^2 \log_2 m)$ a.o. are required for subsequent right-hand sides. The storage requirement is $\mathcal{O}(m^2)$. Also, an iterative solver is considered, where $Px = y$ is solved using a CG-like iterative method. Here P is never explicitly formed. We have proved that the eigenvalues and eigenvectors of P are closely related to the eigendecomposition of $M_{(SC)}^{-1} M_{(BCSC)}$. For a time-dependent and a time-independent model problem, we have established that the spectrum of P is uniformly nonzero.

For constant coefficient PDEs, the SC solver is used as a preconditioner for the restarted GMRES iteration, whereas the BCSC solver yields a direct solution method. For the time-dependent problem, the BCSC solver is a better choice than the SC-preconditioned GMRES method, since the cost for the factorization may be amortized over a large number of time-steps. For the time-independent problem, only one system of equations has to be solved. Hence, the SC-preconditioned GMRES method is the best solution method, since the complexity per unknown is independent of the size of the problem. Comparisons with band Gaussian elimination show that, for large problems, the complexity is higher than those for both the BCSC solver and the SC-preconditioned GMRES method. Also, the storage requirement for band Gaussian elimination becomes prohibitive for large problems.

For the variable coefficient time-dependent problem, both the SC and BCSC solvers are used as preconditioners for the restarted GMRES iteration. For weakly variable coefficients, the solution methods exploiting the SC and BCSC solvers retain a complexity (per unknown) which is independent of problem size. Hence, both methods are faster than Gaussian elimination for large problems. The BCSC solver yields a better approximation of B^{-1} than the SC solver, and it requires less total work. For problems with strongly variable coefficients, the favorable convergence properties are no longer preserved, and the Gaussian elimination solver has the lowest complexity for all problem sizes studied.

Finally, we have used the BCSC solver combined with a semi-implicit time-marching scheme in the solution of the isentropic, almost incompressible Navier-Stokes equations for a driven cavity problem. In [Stoor90] the arising system of equations is solved by a general sparse solver. The two solution methods are compared, and we find that employing the BCSC solver makes it possible to solve a 16 times larger problem than by using the general sparse solver. Also, the CPU-time required is less for the BCSC solver, both in the factorization and the time-marching.

REFERENCES

[Ashby87] S. F. Ashby, *Polynomial Preconditioning for Conjugate Gradient Methods*, Report UIUCDCS-R-87-1355, Dept. of Computer Science, Univ. of Illinois at Urbana-Champaign, Urbana, IL, 1987.

[FrGoNa92] R. W. Freund, G. H. Golub, and N. M. Nachtigal, *Iterative solution of linear systems*, in Acta Numerica, Cambridge University Press, London, 1992, pp. 57–100.

[GuSt91] B. Gustafsson and H. Stoor, *Navier-Stokes equations for almost incompressible flow*, SIAM J. Numer. Anal., 28 (1991), pp. 1523-1547.

[Holm92] S. Holmgren, *CG-Like Iterative Methods and Semicirculant Preconditioners on Vector and Parallel Computers*, Report 148, Dept. of Scientific Computing, Uppsala Univ., Uppsala, Sweden, 1992.

[HoOtto90] S. Holmgren and K. Otto, *A Comparison of Preconditioned Iterative Methods for Nonsymmetric Block-Tridiagonal Systems of Equations*, Report 123 (revised), Dept. of Scientific Computing, Uppsala Univ., Uppsala, Sweden, 1990.

[HoOtto92] ———, *Iterative solution methods and preconditioners for block-tridiagonal systems of equations*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 863-886.

[HoOtto93] ———, *Semicirculant Solvers and Boundary Corrections for First-Order PDE*, Report 149, Dept. of Scientific Computing, Uppsala Univ., Uppsala, Sweden, 1993.

[HoOtto94] ———, *Semicirculant preconditioners for first-order partial differential equations*, SIAM J. Sci. Comput., 15 (1994), pp. 385-407.

[KeyGr87] D. E. Keyes and W. D. Gropp, *A comparison of domain decomposition techniques for elliptic partial differential equations and their parallel implementation*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 166-202.

[VLoan92] C. F. Van Loan, *Computational Frameworks for the Fast Fourier Transform*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.

[Otto97] K. Otto, *Analysis of preconditioners for hyperbolic partial differential equations*, SIAM J. Numer. Anal., 34 (1997), to appear.

[SaadSch86] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856-869.

[Stoor90] H. Stoor, *Numerical Solution of the Navier-Stokes Equations for Small Mach Numbers*, Ph.D. thesis, Dept. of Scientific Computing, Uppsala Univ., Uppsala, Sweden, 1990.

[VdVo82] H. A. Van der Vorst, *Preconditioning by Incomplete Decompositions*, Ph.D. thesis, Rijksuniversiteit Utrecht, Utrecht, the Netherlands, 1982.

SIAM J. SCI. COMPUT.
Vol. 17, No. 3, pp. 631-646, May 1996

004

THE SOLUTION OF MULTIDIMENSIONAL REAL HELMHOLTZ EQUATIONS ON SPARSE GRIDS*

ROBERT BALDER† AND CHRISTOPH ZENGER†

Abstract. Sparse grids provide a very efficient method for the multilinear approximation of functions, especially in higher-dimensional spaces. In the d-dimensional space, the nodal multilinear basis on a grid with mesh size $h = 2^{-n}$ consists of $O(2^{nd})$ basis functions and leads to an L_2-error of order $O(4^{-n})$ and an H_1-error of order $O(2^{-n})$. With sparse grids we get an L_2-error of order $O(4^{-n}n^{d-1})$ and an H_1-error of order $O(2^{-n})$ with only $O(2^n n^{d-1})$ basis functions, if the function u fulfills the condition $\frac{\partial^{2d}}{\partial x_1^2 \partial x_2^2 ... \partial x_d^2} u < \infty$. Therefore, we can achieve much more accurate approximations with the same amount of storage.

A data structure for the sparse grid representation of functions defined on cubes of arbitrary dimension and a finite element approach for the Helmholtz equation with sparse grid functions are introduced. Special emphasis is taken in the development of an efficient algorithm for the multiplication with the stiffness matrix. With an appropriate preconditioned conjugate gradient method (cg-method), the linear systems can be solved efficiently. Numerical experiments are presented for Helmholtz equations and eigenvalue problems for the Laplacian in two and three dimensions, and for a six-dimensional Poisson problem. The results support the assertion that the L_2-error bounds for the sparse-grid approximation are also valid for sparse grid finite element solutions of elliptic differential equations. Problems with nonsmooth solutions are treated with adaptive sparse grids.

Key words. sparse grids, hierarchical finite elements, Helmholtz equation

AMS subject classifications. 65N30, 65N50, 65N55

1. Introduction.

The hierarchical basis, first introduced by Faber [5] in 1909, has been used by Yserentant [12] in 1986 as a preconditioner for the solution of large linear systems arising from elliptic differential equations. Four years later, Zenger [13] showed that a multilinear approximation of a smooth multivariate function u is represented much more efficiently if we directly use a suitably chosen hierarchical basis instead of a nodal basis. This approach leads to the idea of *sparse grids*. Considering a nodal multilinear basis on a grid with uniform mesh size $h = 2^{-n}$ on a d-dimensional cube, which consists of $O(2^{nd})$ functions, then the approximation error in the L_2-norm is of order $O(4^{-n})$, in the H_1-norm of order $O(2^{-n})$. By performing a basis transformation to the hierarchical basis, an L_2-error of order $O(4^{-n}n^{d-1})$ can be obtained with a subset consisting of only $O(2^n n^{d-1})$ transformed basis functions if the function u fulfills the condition $\frac{\partial^{2d}}{\partial x_1^2 \partial x_2^2 ... \partial x_d^2} u < \infty$. If the error in the H_1-norm remains $O(2^{-n})$, we have no decrease in the order here. The application of this concept to the solution of partial differential equations on sparse grids is outlined in the papers by Zenger [13, 14, 15], Bungartz [2, 3], and Griebel [7, 8]. Data structures and algorithms for the solution of Poisson's equation on a square and a cube are presented in [3].

In this paper, we introduce a modified data structure and a sparse grid algorithm that solves Dirichlet problems for the real Helmholtz equation $-\Delta u + cu = f$ on cubes of arbitrary dimension.[1] In §2 we explain the basic characteristics of sparse grids. For a brief theoretical description, we refer to the paper of Bungartz [3]. In §3 a tree structure for the representation of sparse grid functions is introduced. Section 4 deals with a finite element approach for the Helmholtz equation with hierarchical basis functions. It turns out that the number of nonzero entries in the corresponding stiffness matrix is of order $O(4^n n^{d-2})$ for $d \geq 2$. Nevertheless, it is possible to perform a matrix-vector multiplication with $O(2^n n^{d-1})$ operations. The

*Received by the editors April 12, 1993; accepted for publication (in revised form) February 23, 1995.

†Institut für Informatik, Technische Universität München, Arcisstrasse 21, D-80290 München, Germany (balder@informatik.tu-muenchen.de, zenger@informatik.tu-muenchen.de).

[1] We remind the reader that the description of an n-electron system in quantum mechanics leads to a $3n$-dimensional Schrödinger equation.

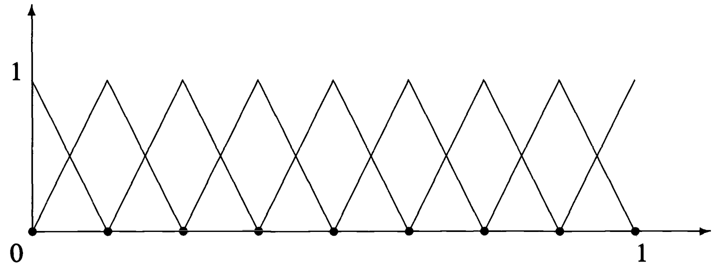


FIG. 1. *Standard basis B_3.*

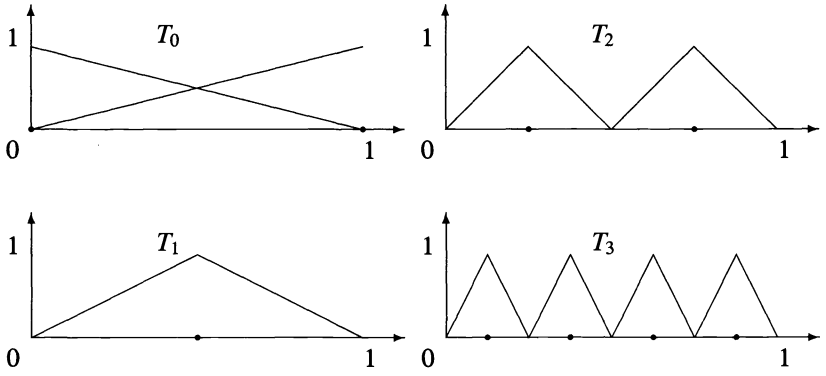


FIG. 2. *Hierarchical basis H_3.*

number of operations is proportional to the size of the vector of unknowns. We prove this constructively in §5 by developing a corresponding algorithm. With the preconditioned cg-method described in Griebel and Oswald [9], we can solve the linear systems efficiently. In §6, we present numerical examples for the solution of the Helmholtz equation with Helmholtz numbers $c \in [-1000, 1000]$ in the unit square and the unit cube. For $c \ll 0$ the corresponding stiffness matrices are indefinite. Nevertheless, we obtain good approximations of the solution of the linear system with the cg-method. Eigenvalue problems with reentrant corners in two and three dimensions are solved with adaptive sparse grids. Finally we look into a six-dimensional Poisson problem. Concluding remarks are made in §7.

2. Sparse grids. Beginning with the one-dimensional case, let S_n be the $(2^n + 1)$-dimensional finite element space of piecewise linear functions on the unit interval on an equidistant grid with mesh size $h = 2^{-n}$. For an arbitrary but fixed n, we consider two different basis representations of this space.

1. The standard basis B_n (see Fig. 1) is the set of piecewise linear functions with the value 1 on one grid point and the value zero on all others.

2. Following Yserentant [12], the hierarchical basis H_n may be defined inductively by

$$H_0 := T_0 := B_0,$$

$$H_i := H_{i-1} \cup T_i = \bigcup_{j=0}^{i} T_j,$$

where T_i, $i \geq 1$, is the set of basis functions of B_i vanishing on the grid belonging to S_{i-1}. This is illustrated in Fig. 2.

The grid point, where a basis function v has the value 1, will be called basis point $bp(v)$. The set of basis points for a given n will be called grid G_n.

Now let $u : [0, 1] \to \mathbb{R}$ be a function with existing and bounded seminorm

$$\mid u \mid_\infty := \left\| \frac{\partial^2}{\partial x^2} u \right\|_\infty \leq C < \infty,$$

and let $u_n \in S_n$ be the linear interpolant of u in the grid G_n. In the standard basis $\{v_i\}$, we define the weights g_i by

$$u_n = \sum_{v_i \in B_n} g_i v_i.$$

Obviously, g_i is just the value of u at the corresponding grid point.

In the hierarchical basis $\{\tilde{v}_i\}$, we define the weights $\tilde{g}_i$ by

$$u_n = \sum_{\tilde{v}_i \in H_n} \tilde{g}_i \tilde{v}_i.$$

The weight $\tilde{g}_i$ for the basis function $\tilde{v}_i \in T_k$ is the difference between the value of the function and the value of the interpolant of H_{k-1} at $bp(\tilde{v}_i)$. Therefore, we call $\tilde{g}_i$ a *hierarchical surplus*. It is easy to show (see [3]) that

$$(1) \qquad \mid \tilde{g}_i \mid \leq \frac{1}{2} 4^{-k} \mid u \mid_\infty \quad \text{for } \tilde{v}_i \in T_k,$$

which means that the weights of the basis functions in T_n decrease geometrically with $n \to \infty$ when $\mid u \mid_\infty$ is bounded. This is precisely the main advantage of the hierarchical over the standard basis and leads to essential savings of storage in the multivariate case. In §5.1 we will show how the hierarchical surpluses can be computed.

The generalization of the standard and the hierarchical bases to higher-dimensional multilinear functions $u_n^{(d)}$ on the unit cube $[0, 1]^d$ is derived from a tensor product decomposition as follows.

For the d-dimensional space we define

$$B_n^{(d)} := \left\{ v_I := v_{i_1,\ldots,i_d}(x_1, x_2, \ldots, x_d) = \prod_{j=1}^{d} v_{i_j}(x_j) : v_{i_j} \in B_n \right\},$$

$$H_n^{(d)} := \left\{ \tilde{v}_I := \tilde{v}_{i_1,\ldots,i_d}(x_1, x_2, \ldots, x_d) = \prod_{j=1}^{d} \tilde{v}_{i_j}(x_j) : \tilde{v}_{i_j} \in H_n \right\}.$$

An upper case index I denotes a multiindex $i_1, \ldots, i_d$. An analogous relation to (1) holds in the multivariate case. Generally, we can state this observation as the following: *The weight of a basis function decreases asymptotically with the square of the volume of its support.*

To be more specific, let $u_n^{(d)}$ be the multilinear interpolant of a function u defined on $[0, 1]^d$ with the seminorm

$$(2) \qquad \mid u \mid_\infty := \left\| \frac{\partial^{2d}}{\partial x_1^2 \partial x_2^2 \ldots \partial x_d^2} u \right\|_\infty < \infty.$$

We define the $\tilde{g}_I$ implicitly by $u_n^{(d)} = \sum_{\tilde{v}_I \in H_n^{(d)}} \tilde{g}_I \tilde{v}_I$. As defined above, we have $\tilde{v}_I = \prod_{j=1}^{d} \tilde{v}_{i_j}(x_j)$, where $\tilde{v}_{i_j} \in T_{k_j}$. Recall that 2^{-k_j} is the length of the support of $\tilde{v}_{i_j} \in T_{k_j}$. With

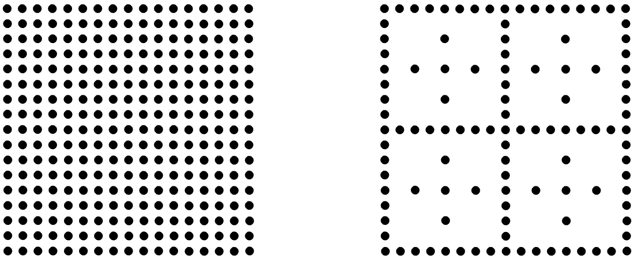


FIG. 3. *Full grid $G_4^{(2)}$ and regular sparse grid $\tilde{G}_4^{(2)}$.*

$k := \sum_{j=1}^{d} k_j$, the bound

$$| \tilde{g}_I | \leq \frac{1}{2^d} 4^{-k} | u |_\infty \tag{3}$$

holds (see [3]).

The basic idea of a sparse grid is as follows: We approximate a function with those functions $\tilde{v}_I$ from the set $H_\infty^{(d)} := \bigcup_{n \in \mathbb{N}} H_n^{(d)}$ where $\tilde{g}_I$ satisfies

$$| \tilde{g}_I | > \varepsilon \tag{4}$$

for a given $\varepsilon > 0$ (see [13]).

For an a priori estimation of $| \tilde{g}_I |$, we assume that $| \frac{\partial^{2d}}{\partial x_1^2 \partial x_2^2 \ldots \partial x_d^2} u |$ is bounded above uniformly in $[0, 1]^d$. Then (3) leads us to consider all basis functions whose support has a volume larger than a given minimal volume. This reduced *sparse* basis is called $\tilde{H}_n^{(d)}$. Note that $\tilde{H}_n^{(d)} \subset H_n^{(d)}$ and $\tilde{H}_{n+1}^{(d)} \not\subset H_n^{(d)}$. The set of the basis points of $\tilde{H}_n^{(d)}$ is exactly our *regular sparse grid* $\tilde{G}_n^{(d)}$ (see Fig. 3), in contrast to an *adaptive sparse grid* which will be defined later in this section.

The space spanned by $\tilde{H}_n^{(d)}$ is called $\tilde{S}_n^{(d)}$.

Bungartz showed in [3] that the number of basis functions in the d-dimensional case is reduced dramatically from $O(2^{nd})$ to $O(2^n n^{d-1})$, while the accuracy of the approximation degrades only from $O(4^{-n})$ to $O(4^{-n} n^{d-1})$ in the L_2-norm. We see that with a slight loss in accuracy we can drastically reduce the dimension of the approximation space and therefore the amount of required storage.

The numerical examples in §6 support our assumption that the previous error bounds are also valid for the solutions of linear elliptic boundary value problems. Multilevel preconditioners [9] assure fast convergence for iterative solvers.

For functions u where $\frac{\partial^{2d}}{\partial x_1^2 \partial x_2^2 \ldots \partial x_d^2} u$ is not bounded, we may not get satisfying results with regular sparse grids. We have to replace our a priori estimation for $| \tilde{g}_I |$ by an a posteriori estimation, say by computed values of $| \tilde{g}_I |$. This approach leads to adaptive sparse grids. Our goal is to approximate the function u by all g_I that fulfill (4). Therefore, it seems natural to refine the grid in every grid point where (4) holds. Refinement means that the basis functions corresponding to one half of the support of v_I should also belong to our grid. Asymptotically, this fulfills the basic idea of a sparse grid. Some numerical experiments with this strategy are presented in §6.2.

3. A data structure for sparse grids. In this section, we describe a data structure well suited for the representation of sparse grid functions in spaces of general dimension and for the algorithms working on it. The data structure consists of a set of trees called *grid trees*.

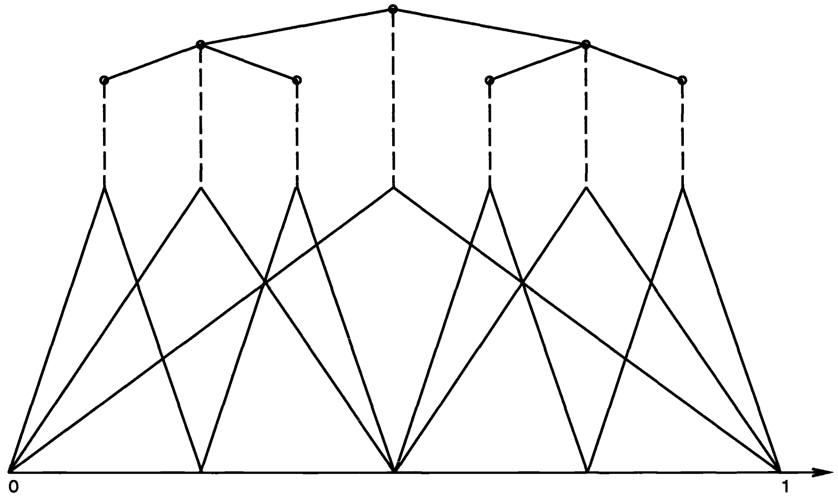


FIG. 4. *Assignment of the basis functions of $\tilde{H}_{n,0}$ to the nodes of a grid tree.*

Every basis function is assigned to a node in a grid tree. We define the subset $\tilde{H}^{(d)}_{n,0} \subset \tilde{H}^{(d)}_n$ as follows:

A basis function $\tilde{v}_I = \prod_{j=1}^d \tilde{v}_{i_j} \in \tilde{H}^{(d)}_n$ belongs to $\tilde{H}^{(d)}_{n,0}$ if and only if $\tilde{v}_{i_j} \notin T_0$ for $j = 1, \ldots, d$.

The basis points of the basis functions from $\tilde{H}^{(d)}_{n,0}$ are exactly those located in $\Omega =]0, 1[^d$ and not those on the boundary $\partial\Omega$ of the domain.

In the one-dimensional case, the grid tree is a binary tree. The basis functions of $\tilde{H}_{n,0} := \tilde{H}^{(1)}_{n,0}$ are assigned to the nodes of the grid tree as shown in Fig. 4. We may now define the grid tree for the basis functions of $\tilde{H}^{(d)}_{n,0}$ recursively. We get the d-dimensional grid tree by combining every node of a one-dimensional grid tree with the root of a $(d-1)$-dimensional grid tree. The location of the node belonging to $\prod_{j=1}^d \tilde{v}_{i_j}$ is recursively defined by the location of $\prod_{j=1}^{d-1} \tilde{v}_{i_j}$ in the $(d-1)$-dimensional grid tree that is connected with the node belonging to $\tilde{v}_{i_d}$ in the one-dimensional grid tree.

The remaining basis functions belong to the boundary of the d-dimensional cube that consists of lower-dimensional cubes. Therefore, they are assigned to nodes of lower-dimensional grid trees. A null-dimensional grid tree is a single node. See Fig. 5 for a two-dimensional example of our data structure for a regular sparse grid. New basis functions can be included by adding new nodes in the tree. Hence, the tree structure is also well suited for adaptive grids.

4. The Helmholtz equation. Let us define the sets

$$\{I\} := \{I : v_I \in H^{(d)}_n\}, \qquad \{I_0\} := \{I : v_I \in H^{(d)}_{n,0}\},$$
$$\{i\} := \{i : v_i \in H_n\}, \qquad \{i_0\} := \{i : v_i \in H_{n,0}\}.$$

We consider the Dirichlet boundary value problem

$$-\Delta u + c \cdot u = f \tag{5}$$

in $\Omega = [0, 1]^d$ with $u(x) = u_0(x)$ on $\partial\Omega$. With the finite-element method, we want an approximation $\tilde{u} \in \tilde{S}^{(d)}_n$ of u, which we obtain by solving the weak form of (5) given by

$$\int_\Omega \nabla\tilde{u} \cdot \nabla\tilde{v}_K + c \cdot \tilde{u} \cdot \tilde{v}_K \, dx = \int_\Omega \tilde{f} \cdot \tilde{v}_K \, dx, \qquad K \in \{I_0\}, \tag{6}$$

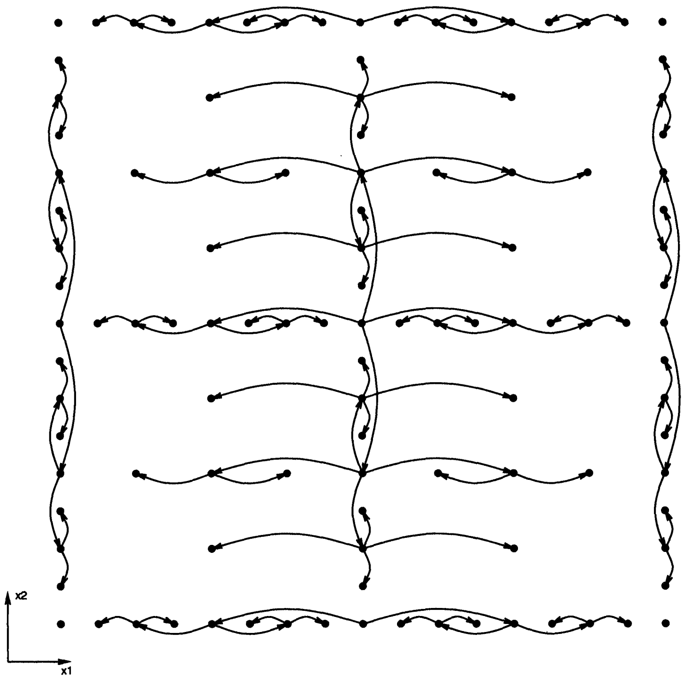


FIG. 5. *Data structure for* $\tilde{H}_4^{(2)}$.

where $\tilde{u} = \sum_{\{I\}} \tilde{g}_I \cdot \tilde{v}_I$ interpolates the function u_0 on $\partial\Omega \cap \tilde{G}_n^{(d)}$ and $\tilde{f} = \sum_{\{I\}} \tilde{f}_I \cdot \tilde{v}_I$ interpolates f in $\tilde{G}_n^{(d)}$.

We define the stiffness matrix

$$(A_{IK}) := \left(\int_\Omega \nabla\tilde{v}_I \cdot \nabla\tilde{v}_K + c \cdot \tilde{v}_I \cdot \tilde{v}_K \, dx \right), \qquad I, K \in \{I_0\},$$

the mass matrix

$$(M_{IK}) := \left(\int_\Omega \tilde{v}_I \cdot \tilde{v}_K \, dx \right), \qquad I, K \in \{I_0\},$$

and the corresponding extended rectangular matrices $(\bar{A}_{IK})$ and $(\bar{M}_{IK})$ resulting from the definitions above if $I \in \{I\}$. With the coefficient vectors $(\tilde{g}_I)$ and $(\tilde{f}_I)$, we may write (6) as

$$(\bar{A}_{IK})(\tilde{g}_I) = (\bar{M}_{IK})(\tilde{f}_I).$$

With $\tilde{v}_I = \prod_{l=1}^d \tilde{v}_{i_l}(x_l)$, the entries of the mass matrix are

$$M_{IK} = \int_\Omega \prod_{l=1}^d \tilde{v}_{i_l}(x_l) \cdot \prod_{l=1}^d \tilde{v}_{k_l}(x_l) \, dx = \prod_{l=1}^d \int_0^1 \tilde{v}_{i_l}(x_l) \cdot \tilde{v}_{k_l}(x_l) \, dx. \tag{7}$$

For the Laplacian part of the stiffness matrix, we have

$$\begin{aligned} \int_\Omega \nabla\tilde{v}_I \cdot \nabla\tilde{v}_K \, dx &= \int_\Omega \left(\nabla \prod_{l=1}^d \tilde{v}_{i_l}(x_l) \right) \cdot \left(\nabla \prod_{l=1}^d \tilde{v}_{k_l}(x_l) \right) dx \\ &= \sum_{j=1}^d \int_\Omega \frac{\partial}{\partial x_j} \prod_{l=1}^d \tilde{v}_{i_l}(x_l) \cdot \frac{\partial}{\partial x_j} \prod_{l=1}^d \tilde{v}_{k_l}(x_l) \, dx. \end{aligned} \tag{8}$$

Let us consider the first summand ($j = 1$) of this term. Due to the hierarchical structure of the basis functions, all parts of the sum are zero when $\tilde{v}_{i_1}$ is not equal to $\tilde{v}_{k_1}$. For $\tilde{v}_{i_1} = \tilde{v}_{k_1}$ we get

$$\int_0^1 \frac{\partial}{\partial x_1}\tilde{v}_{i_1}(x_1) \cdot \frac{\partial}{\partial x_1}\tilde{v}_{k_1}(x_1)\,dx_1 = \frac{2}{h_{k_1}}, \tag{9}$$

where h_{k_1} is half of the mesh size of the support of $\tilde{v}_{k_1}$. Hence, we have

$$\int_\Omega \frac{\partial}{\partial x_1}\tilde{v}_I \cdot \frac{\partial}{\partial x_1}\tilde{v}_K\,dx = \delta_{i_1k_1}\frac{2}{h_{k_1}}\prod_{l=2}^{d}\int_0^1 \tilde{v}_{i_l}(x_l) \cdot \tilde{v}_{k_l}(x_l)\,dx_l, \tag{10}$$

where $\delta_{i_1k_1}$ denotes the well-known Kronecker symbol. Apart from the number of multiplicands and the factor $\delta_{i_1k_1}\frac{2}{h_{k_1}}$, this term is analogous to the right-hand side of (7). For the other summands in (8), we get similar formulas.

We will now estimate the number of entries in the stiffness matrix. From (7) and (10), we see that

$$\int_\Omega \nabla\tilde{v}_I\nabla\tilde{v}_K\,dx \neq 0 \Rightarrow \int_\Omega \tilde{v}_I\tilde{v}_K \neq 0.$$

Therefore, if the Helmholtz coefficient c is nonzero, then the number of nonzero entries in the stiffness matrix is equal to the number of nonzero entries in the mass matrix (M_{IK}).

Let $\tilde{w}$ be the basis function from T_1, $d \geq 2$ and the sets $\mathcal{S}_1$ and $\mathcal{S}_2$ be defined as

$$\mathcal{S}_1 := \{\tilde{v}_I \in H_{n,0}^{(d)} : \tilde{v}_{i_d} = \tilde{w}\}, \qquad \mathcal{S}_2 := \{\tilde{v}_K \in H_{n,0}^{(d)} : \tilde{v}_{k_1} = \tilde{v}_{k_2} = \cdots = \tilde{v}_{k_{d-1}} = \tilde{w}\}.$$

We have card$(\mathcal{S}_1) = O(2^n \cdot n^{d-2})$ and card$(\mathcal{S}_2) = O(2^n)$, where card(.) denotes the cardinality of a set. The support of a function v is defined as the open set $\{x : v(x) \neq 0\}$. Then support$(\tilde{w}) =]0, 1[$ and from support$(\tilde{v}_i) \cap$ support$(\tilde{w}) \neq \emptyset\, \forall i \in \{i\}$, we obtain

$$v_I \in \mathcal{S}_1, v_K \in \mathcal{S}_2 \Rightarrow \text{support}(\tilde{v}_I) \cap \text{support}(\tilde{v}_K) \neq \emptyset.$$

The corresponding M_{IK} is positive because the basis functions are positive on their support. This means that we have at least card$(\mathcal{S}_1)\cdot$card$(\mathcal{S}_2) = O(4^n \cdot n^{d-2})$ nonzero entries in the mass matrix and the stiffness matrix, respectively. Nevertheless, as we will see in the next section, it is possible to perform a matrix-vector multiplication with $O(2^n \cdot n^{d-1})$ operations.

5. Algorithm. In this section we develop an algorithm that performs a matrix-vector multiplication with the stiffness matrix with a constant amount of operations per vector component. We first introduce the basis transformations between hierarchical and standard basis, called the *hierarchical transformations*. Then, we show how to perform a multiplication with the mass matrices in the one-dimensional case. Finally, we reduce the d-dimensional matrix multiplication recursively to the one-dimensional case.

5.1. The hierarchical transformations. To describe these transformations, we make the following definition.

The hierarchical neighbors of an index $i \in \{i_0\}$ are the two indices $l(i)$ and $r(i)$, $l, r : \{i_0\} \to \{i\}$, where the basis points $bp(\tilde{v}_{l(i)})$ and $bp(\tilde{v}_{r(i)})$ are located at the two ends of the support of $\tilde{v}_i$ (see Fig. 6).

Let us study the basis transformations in S_n between the hierarchical basis H_n and the standard basis B_n. If and only if their basis points are identical shall $\tilde{v}_i \in H_n$ and $v_i \in B_n$

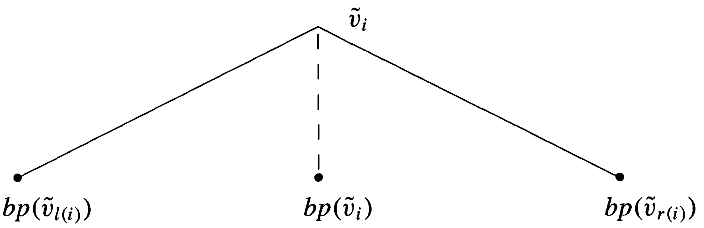


FIG. 6. *Hierarchical neighbors.*

have the same index i. Furthermore, the set $\{1, 2, \ldots, \text{card}(H_n)\}$ of indices should fulfill the conditions $i > l(i)$ and $i > r(i)$ for $i \in \{i_0\}$. Obviously the basis functions from T_0 have the indices 1 and 2, and the basis function $\tilde{w}$ from T_1 has the index 3. From the definition of the hierarchical basis, we have the relations

$$\tag{11} \begin{array}{ll} \tilde{g}_i = g_i - \frac{1}{2}(g_{l(i)} + g_{r(i)}), & i \in \{i_0\}, \\ \tilde{g}_i = g_i, & \{i : v_i \in T_0\}. \end{array}$$

Consequently, the matrix $\bar{Y} = (\bar{Y}_{ij})$ that performs the transformation from the standard to the hierarchical basis is given by

$$\bar{Y}_{ij} = \begin{cases} -\frac{1}{2} : j \in \{l(i), r(i)\}, \\ \delta_{ij} : \text{ otherwise.} \end{cases}$$

We define the matrix Y to be the submatrix of $\bar{Y}$, whose first two rows and columns corresponding to the basis functions from T_0 are missing. Because of the conditions above and (11), $\bar{Y}$ and Y are lower triangular matrices with at most three entries per line. We call these transformations *hierarchical transformations*. The inverse hierarchical transformations $\bar{Y}^{-1}$ and Y^{-1}, called *inverse transformations*, are obviously computable by backward substitution. The corresponding relations for $\bar{Y}^{-1}$ are

$$\tag{12} \begin{array}{ll} g_i = \tilde{g}_i + \frac{1}{2}(g_{l(i)} + g_{r(i)}), & i \in \{i_0\}, \\ g_i = \tilde{g}_i, & \{i : v_i \in T_0\}. \end{array}$$

Note that all these transformations can be performed with a constant amount of work per vector component.

5.2. The one-dimensional case. Here we derive a special decomposition of the mass matrices $(M_{i,k})$ and $(\bar{M}_{i,k})$. Define the matrix D by

$$D := (D_{i,k}) := (h_i \cdot \delta_{ik}), \qquad i, k \in \{i_0\}$$

and the corresponding rectangular matrix $\bar{D}$ with $i \in \{i\}$. The identity matrix is denoted by E. We distinguish the following three cases:

1. $i = k$, $\qquad \int_0^1 \tilde{v}_i(x) \cdot \tilde{v}_i(x)\, dx = \frac{2}{3} \cdot h_i$.

The "diagonal part" of the mass matrices may be written as $\frac{2}{3}D$ and $\frac{2}{3}\bar{D}$, respectively.

2. support$(\tilde{v}_i) \supset$ support$(\tilde{v}_k)$ denoted by $\{i \supset k\}$.

This corresponds to the "lower triangular part" of the mass matrices because it follows from this condition that $i < k$. We consider $(\bar{M}_{i,k})$ first. Obviously, $\sum_{\{i \supset k\}} \tilde{g}_i\, \tilde{v}_i(x)$ is linear on the support of $\tilde{v}_k$, and therefore is determined there by the absolute values $g_{l(k)}$ and $g_{r(k)}$. With that, we obtain

$$\int_0^1 \sum_{\{i\supset k\}} \tilde{g}_i\, \tilde{v}_i\, \tilde{v}_k\, dx = h_k \frac{g_{l(k)} + g_{r(k)}}{2} = h_k \left(\tilde{g}_k + \frac{g_{l(k)} + g_{r(k)}}{2} - \tilde{g}_k \right). \tag{13}$$

With (12), we easily verify that this corresponds exactly to the matrix $D(\bar{Y}^{-1} - E)$. For the lower triangular part of $(M_{i,k})$, we just have to leave out the first and second columns and get $D(Y^{-1} - E)$.

3. support$(\tilde{v}_i) \subset$ support$(\tilde{v}_k)$ denoted by $\{i \subset k\}$.

For $(M_{i,k})$ our approach is symmetric in $\tilde{v}_i$ and $\tilde{v}_k$. Here, we get the transpose of the matrix from case 2, namely $(Y^{-T} - E)D$. Because $i > k$, this is the upper triangular part of $(M_{i,k})$ and the entries are identical to the upper triangular part of $(\bar{M}_{i,k})$. For the latter, we may write $(Y^{-T} - E)\bar{D}$.

With that, the mass matrices in the one-dimensional case are

$$(M_{ik}) = D(Y^{-1} - E) + (Y^{-T} - E)D + \frac{2}{3}D,$$

$$(\bar{M}_{ik}) = D(\bar{Y}^{-1} - E) + (Y^{-T} - E)\bar{D} + \frac{2}{3}\bar{D}.$$

Note that Y^T is an upper triangular matrix and that the costs for backward substitution for computing a matrix multiplication with Y^{-T} are equal to the costs for the inverse transformation. Therefore, the costs are proportional to the number of coefficients.

5.3. Reduction to the one-dimensional case. Our goal is an algorithm for the computation of a mass matrix multiplication

$$(M_{IK})(\tilde{g}_I) = \left(\int_\Omega \sum_{\{I_0\}} \tilde{g}_I\, \tilde{v}_I\, \tilde{v}_K\, dx \right). \tag{14}$$

The problem in general dimensions is reduced to a sequence of multiplications with one-dimensional mass matrices. We explain the method only for the mass matrix (M_{IK}), because the construction of the method for the extended mass matrix is identical. We will see that we have to take special care of the storage of interim results.

Define the four sets

$$\begin{aligned}
\{J\} &:= \{J := (j_1, \dots, j_{d-1}) : \exists j_d : (j_1, \dots, j_{d-1}, j_d) \in \{I_0\}\},\\
\{I_J\} &:= \{I \in \{I_0\} : i_1 = j_1, \dots, i_{d-1} = j_{d-1}\},\\
\{j_d\} &:= \{j_d : \exists J : (j_1, \dots, j_{d-1}, j_d) \in \{I_0\}\},\\
\{I_{j_d}\} &:= \{I \in \{I_0\} : i_d = j_d\}.
\end{aligned}$$

With that, we can write

$$\int_\Omega \sum_{\{I_0\}} \tilde{g}_I\, \tilde{v}_I\, \tilde{v}_K\, dx = \begin{cases} \displaystyle\sum_{\{J\}} \left(\sum_{\{I_J\}} \tilde{g}_I \int_0^1 \tilde{v}_{i_d}\, \tilde{v}_{k_d}\, dx_d \right) \prod_{l=1}^{d-1} \int_0^1 \tilde{v}_{j_l}\, \tilde{v}_{k_l}\, dx_l, \\ \displaystyle\sum_{\{j_d\}} \left(\sum_{\{I_{j_d}\}} \tilde{g}_I \prod_{l=1}^{d-1} \int_0^1 \tilde{v}_{i_l}\, \tilde{v}_{k_l}\, dx_l \right) \int_0^1 \tilde{v}_{j_d}\, \tilde{v}_{k_d}\, dx_d. \end{cases} \tag{15}$$

We will now examine the computation of the two-dimensional integral

$$\int_{[0,1]^2} \sum_{\{I_0\}} \tilde{g}_I\, \tilde{v}_I\, \tilde{v}_K\, dx \tag{16}$$

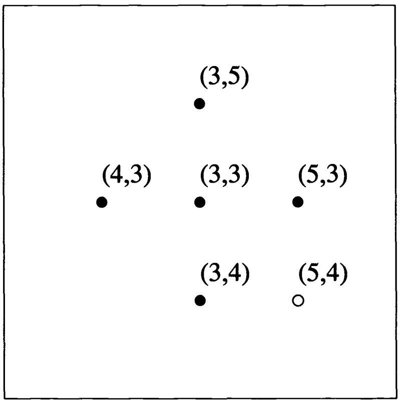


FIG. 7. *The basis points of $\tilde{H}^{(2)}_{2,0}$ in the unit square are marked with a $\bullet$, there is for example no basis function in $\tilde{H}^{(2)}_{2,0}$ that corresponds to the basis point marked with a $\circ$.*

with the upper formula of the right-hand side of (15). We can compute the terms $\sum_{\{I_J\}} \tilde{g}_I \int_0^1 \tilde{v}_{i_2}\, \tilde{v}_{k_2}\, dx_2$ from (15) with the techniques from §5.2. The interim results would be stored in the nodes with multiindex (i_1, k_2). With these coefficients, we could compute $\sum_{\{J\}}(\ldots) \int_0^1 \tilde{v}_{i_1}\, \tilde{v}_{k_1}\, dx_1$. But, if support$(\tilde{v}_{i_2}) \supset$ support$(\tilde{v}_{k_2})$, a node with multiindex (i_1, k_2) might not exist in our sparse grid (see Fig. 7).

We get an analogous effect if we compute (16) by the alternate formula of (15). On the other hand, if support$(\tilde{v}_{i_2}) \subseteq$ support$(\tilde{v}_{k_2})$, a node with multiindex (i_1, k_2) exists by construction of the sparse grid. This leads us to redefine the sets $\{I_J\}$ and $\{I_{j_d}\}$ by

$$\{I_J\} := \{I \in \{I_0\} : i_1 = j_1, \ldots, i_{d-1} = j_{d-1}, \text{support}(\tilde{v}_{i_d}) \subseteq \text{support}(\tilde{v}_{k_d})\},$$
$$\{I_{j_d}\} := \{I \in \{I_0\} : i_d = j_d, \text{support}(\tilde{v}_{i_d}) \supset \text{support}(\tilde{v}_{k_d})\}.$$

Now, we may write

$$\begin{aligned}\int_\Omega \sum_{\{I_0\}} \tilde{g}_I\, \tilde{v}_I\, \tilde{v}_K\, dx &= \sum_{\{J\}} \left[\sum_{\{I_J\}} \tilde{g}_I \int_0^1 \tilde{v}_{i_d}\, \tilde{v}_{k_d}\, dx_d\right] \prod_{l=1}^{d-1} \int_0^1 \tilde{v}_{i_l}\, \tilde{v}_{k_l}\, dx_l \\ &\quad + \sum_{\{j_d\}} \left[\sum_{\{I_{j_d}\}} \tilde{g}_I \prod_{l=1}^{d-1} \int_0^1 \tilde{v}_{i_l}\, \tilde{v}_{k_l}\, dx_l\right] \int_0^1 \tilde{v}_{i_d}\, \tilde{v}_{k_d}\, dx_d.\end{aligned} \tag{17}$$

In the first summand, the term in brackets is a one-dimensional integral; the nodes, where the interim results are stored, exist because of our new definition of $\{I_J\}$. The computed coefficients are the coefficients for $(d-1)$-dimensional problems. In the second summand, the term in brackets is a $(d-1)$-dimensional problem. Solving these problems leads to the coefficients of one-dimensional integrals. We now may apply these splittings recursively to the $(d-1)$-dimensional problems, and thus get directly solvable one-dimensional problems. Due to the splittings, all interim results are stored in existing nodes.

We now have described how to compute a term $\int_\Omega \sum_{\{I_0\}} \tilde{g}_I\, \tilde{v}_I\, \tilde{v}_K\, dx$ for a fixed K. For the computation of the vector $(\int_\Omega \sum_{\{I_0\}} \tilde{g}_I\, \tilde{v}_I\, \tilde{v}_K\, dx)$, note that the splitting in (17) corresponds to a splitting of the one-dimensional mass matrices into the two parts $(Y^{-T} - E)D + \frac{2}{3}D$ and $D(Y^{-1} - E)$. Therefore, the one-dimensional integrals can be computed by multiplication with one of the partial matrices. Considering the operations that have to be made for a matrix multiplication (e.g., relation (12) or (13)) we find that they can be computed separately for each node, similar to the computation of finite difference stars. Therefore we do not compute these matrices explicitly but replace them by procedures that traverse our data structure and perform the operations in every node. In particular, the treatment of adaptive grids is then transferred to the problem of traversing an adaptive data structure which is very simple in our case (tree structure). We find descriptions of such procedures in Balder [1].

The amount of work for the computation of the set of one-dimensional integrals is proportional to the overall number of coefficients. If we denote the number of those computations for a d-dimensional problem by $C(d)$, we get from (17) that $C(1) = 2$ and $C(l+1) = 2+2\,C(l)$. It is easy to prove by induction that $C(d) = 2^{d+1} - 2$. From (10), we see that we can use a similar algorithm for the computation of the components of the Laplacian, and, therefore, can also compute a multiplication with the stiffness matrix. Note that the amount of work for this multiplication is proportional to the number of coefficients $O(n \log(n)^{d-1})$ and has a better complexity than the number of nonzero entries in the stiffness matrix $O(n^2 \log(n)^{d-2})$. With an implementation that is optimized, with respect to storage we need for the matrix multiplication

$$OP(d) = 17\left(\left(2^d - 1\right) + (d-1)\left(2^{d-1} - 1\right)\right)$$

operations (*add* or *mult*) for every coefficient; for example, $OP(2) = 68$ and $OP(3) = 221$.

6. Numerical results. The following numerical experiments have been computed on workstations with up to 128 megabytes of main storage. The programs are written in the C++ programming language. The linear equations are solved with a BPX-like multilevel preconditioned cg-method which is described in Griebel and Oswald [9]. It uses additional residuals corresponding to functions $v_S \in \tilde{S}_n^{(d)}$ that can easily be computed by linear combination of residuals corresponding to $v_K \in \tilde{H}_n^{(d)}$. The amount of work is about $PCOP(d) = 8\left(2^d - 1\right)$ operations for every coefficient ($PCOP(2) = 24, PCOP(3) = 56$) for the computation of the additional residuals. The extended vector of residuals is about 2^d times larger than the original vector. This causes a corresponding additional amount of work for the computation of the scalar products needed for the cg-method. The preconditioner is designed for the two-dimensional Poisson equation and proved to be optimal in this case. The generalization for higher dimensions is straightforward. It also works well for the Helmholtz equation. In this paper we will not discuss this preconditioner but present convergence factors that demonstrate that the equations can be solved efficiently.

6.1. Helmholtz equation. Our first test problem is given by the equation

$$-\Delta u + c \cdot u = c \cdot w, \qquad u = 0 \text{ on } \partial\Omega,\ \Omega = [0,1]^d, \tag{18}$$

where $w = \prod_{l=1}^{d} \tilde{w}(x_l)$ and $\tilde{w}$ is the basis function from T_1.

Let u_I be the value of the function u at the node with multiindex I and $\{I_n^{(d)}\} := \{I : v_I \in H_n^{(d)}\}$. Note that $\{I_n^{(d)}\}$ corresponds to a full grid $G_n^{(d)}$. Define a discrete L_2-norm of a continuous function u by

$$\mathcal{L}(u) := \sum_{\{I_n^{(d)}\}} (u_I)^2. \tag{19}$$

We will examine here the error $\mathcal{E}$ of the numerical solution defined by

$$\mathcal{E} := \mathcal{L}(u - \tilde{u})/\mathcal{L}(u), \tag{20}$$

where $\tilde{u}$ is the numerical approximation and u the *exact* solution of (18). $\mathcal{E}$ is the discrete L_2-norm of the error scaled by the discrete L_2-norm of the solution. For this problem,

TABLE 1
Helmholtz equation.

Dimension 2							
n	unknowns	$c = 1000$	$c = 100$	$c = 10$	$c = -10$	$c = -100$	$c = -1000$
5	257	$1.3E-03$	$8.7E-04$	$8.7E-04$	$1.8E-03$	$1.2E+00$	$7.3E-02$
6	577	$3.7E-04$	$2.4E-04$	$2.3E-04$	$4.4E-04$	$1.6E-01$	$3.3E-02$
7	1281	$1.2E-04$	$7.5E-05$	$6.2E-05$	$1.0E-04$	$3.6E-02$	$2.6E-02$
8	2817	$3.6E-05$	$2.5E-05$	$1.8E-05$	$8.0E-06$	$8.8E-03$	$1.6E-02$
9	6145	$9.2E-06$	$6.4E-06$	$4.8E-06$	$2.2E-06$	$2.2E-03$	$2.6E-03$
10	13313	$2.4E-06$	$1.7E-06$	$1.3E-06$	$6.5E-07$	$5.5E-04$	$6.3E-04$
11	28673	$6.6E-07$	$4.4E-07$	$3.3E-07$	$1.9E-07$	$1.4E-04$	$1.5E-04$
cf		0.50	0.48	0.50	0.49	–	–

Dimension 3							
n	unknowns	$c = 1000$	$c = 100$	$c = 10$	$c = -10$	$c = -100$	$c = -1000$
5	3713	$7.8E-04$	$6.2E-04$	$5.3E-04$	$4.0E-04$	$1.6E-02$	$7.9E-02$
6	8961	$2.1E-04$	$1.7E-04$	$1.5E-04$	$1.2E-04$	$4.0E-03$	$4.9E-02$
7	21249	$6.3E-05$	$5.1E-05$	$4.2E-05$	$3.6E-05$	$1.0E-03$	$1.6E-01$
8	49665	$2.1E-05$	$1.4E-05$	$1.2E-05$	$1.0E-05$	$2.4E-04$	$1.1E-02$
cf		0.70	0.65	0.63	0.63	–	–

we approximate the exact solution by a solution we obtain from a finer grid. In Table 1, we examine a two- and a three-dimensional example. We have chosen $\{I_7^{(2)}\}$ ($\{I_4^{(3)}\}$) in (20) and approximate the exact solution by the numerical solution in the space $\tilde{S}_{14}^{(2)}$ ($\tilde{S}_{10}^{(3)}$) for the two- and three-dimensional case. We have listed the errors $\mathcal{E}$ depending on the approximation space $\tilde{S}_n^{(d)}$ and the corresponding number of unknowns for different Helmholtz numbers. In the last row, we have given the convergence factors cf of the preconditioned cg-method defined by

$$cf := \left(\|r^{5+m} \|_2 \, / \, \| r^5 \|_2\right)^{\frac{1}{m}} . \tag{21}$$

Here r^i is the preconditioned residual after the ith cg-iteration. We choose m so that $\| r^{5+m} \|_2 < 10^{-10} \cdot \| r^5 \|_2$. The value cf is an average decrease factor of the 2-norm of the preconditioned residual, and corresponds to the grids $\tilde{G}_{14}^{(2)}$ and $\tilde{G}_{10}^{(3)}$ respectively. We use an averaged factor because the norm of the residual is not in every case monotonely decreasing during the cg-iteration. Note that the stiffness matrix is indefinite if $c < ev \approx -d \cdot \pi^2$. We give no convergence factors for these problems because the cg-method is not convergent. On the other hand we observe a significant reduction of the 2-norm of the residual at the beginning of the cg-iteration. This leads to an error in the solution of the linear system which is small compared with the approximation error. The solution is therefore good enough for our investigations. One may use, for example, the method of conjugate residuals (e.g., Hackbusch [10]) to obtain the exact solution for these indefinite problems.

The errors behave as we could expect from the theoretical results. Asymptotically, the factor of error reduction with respect to n reaches the value 4. This confirms the $O(4^{-n}n^{-d})$ accuracy in the L_2-norm mentioned in §2. In Fig. 8, we compare how the error $\mathcal{E}$ depends on the number of unknowns for sparse and for full grids. We see that the solutions corresponding to sparse grids with about 10000 unknowns are about two orders of magnitude better than the solutions we obtained on full grids.

6.2. Eigenvalues of the Laplacian on adaptive sparse grids. By combining cubes, we may construct more complex domains. We have computed some eigenvalues and eigenfunctions of the Laplacian in the two-dimensional L-domain and its three-dimensional counterpart,

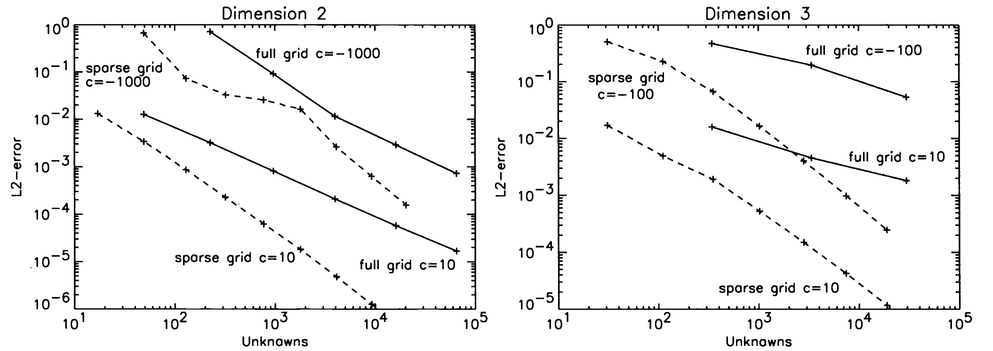


FIG. 8. *Comparison of full and sparse grids.*

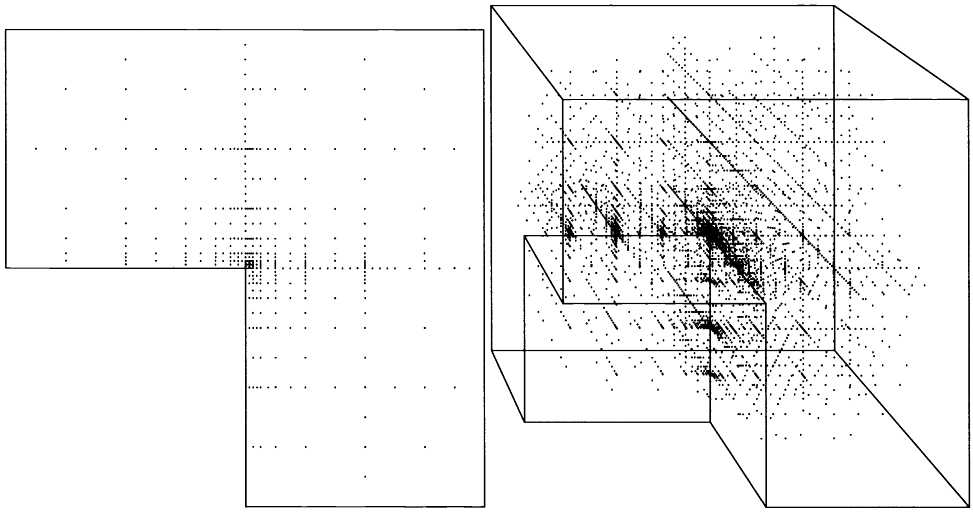


FIG. 9. *Adaptive sparse grids.*

a cube where one octant is removed. The reentrant corners in these domains lead to singularities in the solutions. It is typical for such singularities that the hierarchical surpluses at the singularity decrease asymptotically with a factor less than 4. Therefore, adaptive grids are necessary to obtain cost effective approximations. For our numerical experiments, we use the following strategy. If the hierarchical surplus satisfies $| \tilde{g}_l | > \varepsilon$ for a given $\varepsilon > 0$, then the $2d$ basis functions corresponding to one half of the support of v_l should also belong to our grid. Examples for adaptive sparse grids are given in Fig. 9. Note that the solution is singular along the reentrant edges in the three-dimensional domain. On the other hand, the solution is almost linear along those edges. Therefore, due to the basis functions that have a large support in the direction parallel to the edge and a small support in the other directions, we do not need to have a fine grid along the whole singularity.

To solve the eigenvalue problem we have to find Helmholtz numbers λ, where the Helmholtz equation

$$-\Delta u = \lambda \cdot u \tag{22}$$

with homogeneous boundary conditions has nontrivial solutions. Exact solutions for this problem are not known for the L-shaped domain, but approximate eigenvalues are given in

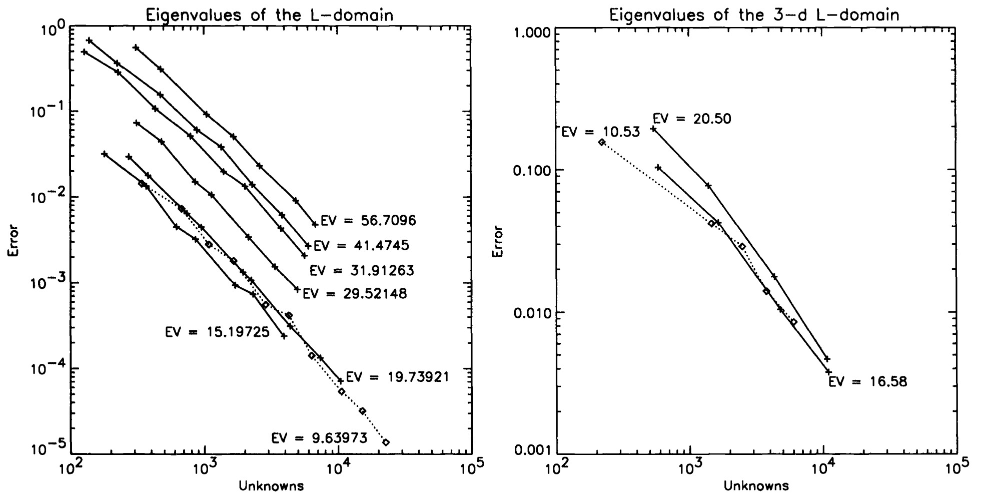


FIG. 10. *Eigenvalue errors.*

Fox, Henrici, and Moler [6]. Discretization of (22) leads to a generalized eigenvalue problem

$$-A(\tilde{g}_l) = \lambda M(\tilde{g}_l), \tag{23}$$

where A is the stiffness matrix for the Laplacian and M the mass matrix. We compute the eigenvalues and eigenvectors with a Wielandt iteration for the generalized problem. With an approximate eigenvector $(g_l)^j$ one gets an improved approximation $(g_l)^{j+1}$ by solving

$$(A - \bar{\lambda} M)(g_l)^{j+1} = \alpha_j M(g_l)^j,$$

where $\bar{\lambda}$ is an approximation of the eigenvalue and α_j a damping factor. We obtain approximations for the eigenvalue λ from the Raleigh-quotient

$$\lambda \approx \frac{(g_l)^{j^T} A(g_l)^j}{(g_l)^{j^T} M(g_l)^j}.$$

This method is described in Peters and Wilkinson [11]. In Fig. 10, we see the absolute errors err of the eigenvalues dependent on the number of unknowns N. The *exact* solutions for the two-dimensional case are taken from [6]. In the three-dimensional case we estimate the exact eigenvalue by the solution on a finer grid. The value $\log(err)$ seems to be a linear function of $\log(N)$ where the gradient of this function does not depend on the eigenvalue. This leads to the following relation between the error err and the number of unknowns N:

$$err \approx c \cdot N^{-k},$$

where $k = 1.6 \pm 0.1$ for $d = 2$ and $k = 1.35 \pm 0.1$ for $d = 3$ for the results presented here.

6.3. A six-dimensional Poisson problem. We solve the problem

$$-\Delta u = 1, \qquad u = 0 \text{ on } \partial\Omega, \ \Omega =]0, 1[^6, \tag{24}$$

with homogeneous boundary conditions. We assume the right-hand side f to be $f = 1$ on Ω and $f = 0$ on $\partial\Omega$ and expand it into a Fourier series. We may solve the differential equation

TABLE 2
A six-dimensional Poisson problem.

Dimension 6		
n	unknowns	$\mathcal{E}$
4	545	$1.03E-01$
5	2561	$5.55E-02$
6	10625	$3.07E-02$
7	40193	$1.98E-02$
8	141569	$1.39E-02$
fd	117649	$7.31E-02$
fe	117649	$7.58E-02$

exactly for each summand of the series. By superposition we obtain the solution u of (24)

$$u = \sum_{i_1,\dots,i_6=0}^{\infty} \alpha_I \prod_{k=1}^{6} \sin((2i_k+1)\pi x_k) \tag{25}$$

with the coefficients

$$\alpha_I := \alpha_{i_1,\dots,i_6} = \frac{4^6}{\pi^8 \prod_{k=1}^{6}(2i_k+1)\sum_{k=1}^{6}(2i_k+1)^2}. \tag{26}$$

With that formula we can compute the exact solution u of (24) with potentially arbitrary accuracy. In Table 2, we have listed the index n corresponding to the approximation space $\tilde{S}_n^{(6)}$, the number of unknowns, and the error $\mathcal{E}$ corresponding to $\{I_3^{(6)}\}$ in (20). In the last two rows we see the number of unknowns and the error we obtain by solving (24) on the full grid $\{G_3^{(6)}\}$ with mesh size $h = \frac{1}{8}$. In the first case we have used a finite difference (fd) scheme (13-point star) and in the second case the usual finite element (fe) approach with multilinear basis functions. There are two reasons why the results for the sparse grid solutions are not as good as in the previous examples. First, the error is of order $O(4^{-n}\, n^5)$ for six-dimensional problems; the term n^5 is not small compared with 4^n for $n = 8$. On the other hand we cannot even expect an error of this order because the solution does not fulfill the condition (2) as one can easily derive from (25) and (26). Nevertheless, the sparse-grid solutions are clearly better than the solution we obtained on a full grid. The error of the sparse-grid solution using 2561 unknowns is smaller than the error of the full-grid solutions using 117649 unknowns. Note that on full grids, if we assume an error of order $O(h^2)$, we need b^3 times more unknowns if we want to reduce the error by a factor b.

The time-independent Schrödinger equation for two electron atoms or molecules leads to an eigenvalue problem of the form $-\Delta\Psi + V(x)\,\Psi = E\,\Psi,\ x \in \mathbb{R}^6$ (see Conroy [4]). The treatment of such equations with sparse grids is a topic of current research.

7. Concluding remarks. Sparse grids are a technique for the efficient representation of multivariate functions. Compared with full grids, the saving in storage is dramatic, especially for higher dimensions [3]. On the other hand we are able to get much better accuracies if the amount of storage is limited. In our opinion, this is the main advantage of sparse grids. This paper has described the discretization and the solution of the Helmholtz equation on domains of general dimension with regular and adaptive sparse grids. It has been proved that one can perform a matrix-vector multiplication with the corresponding stiffness matrix with a constant amount of work per vector coefficient, although the complexity of the number of nonzero entries in the stiffness matrix is worse than the complexity of the number of unknowns. Together with the preconditioner described in [9], this allows us to solve the Helmholtz problem in optimal time. This means in the first case that we can obtain solutions with a reasonable effort. We have not made comparisons with other programs in respect to computing time. One reason for this is that we have designed our code to be as simple and as flexible as possible because we think it is more important to extend the area of applicability of sparse-grid techniques than to gain speed. Another advantage of the sparse-grid approach is adaptivity. Refinement, understood as enlargement of the approximation space, is, due to the hierarchical basis, done by inclusion of additional basis functions and not by replacement.

What remains is to investigate to what extent the sparse-grid approach can be generalized. Topics of current research directly based on this work are the treatment of more general equations, i.e., the Schrödinger equation and time-dependent problems. Other sparse-grid research in Munich is done on higher-order basis functions, general second-order equations on general domains, and higher-order equations like the biharmonic.

REFERENCES

[1] R. Balder, *Adaptive Verfahren für elliptische und parabolische Differentialgleichungen auf dünnen Gittern*, Ph.D. thesis, Technische Universität München, München, Germany, 1994.

[2] H.-J. Bungartz, *An adaptive Poisson solver using hierarchical bases and sparse grids*, in Proceedings of the IMACS International Symposium on Iterative Methods in Linear Algebra, Brüssel, 2.4.-4.4. 1991, P. de Groen and R. Beauwens, eds., Elsevier, Amsterdam, 1992; Technische Universität München, SFB-Bericht Nr. 342/21/91 A.

[3] ———, *Dünne Gitter und deren Anwendung bei der adaptiven Lösung der dreidimensionalen Poisson-Gleichung*, Ph.D. thesis, Technische Universität München, München, Germany, 1992.

[4] H. Conroy, *Molecular Schrödinger Equation. IV. Results for one- and two-electron systems*, J. Chem. Phys., 41 (1964), pp. 1341-1351.

[5] G. Faber, *Über stetige Funktionen*, Math. Ann., 66 (1909), pp. 81-94.

[6] L. Fox, P. Henrici, and C. Moler, *Approximations and bounds for eigenvalues of elliptic operators*, SIAM J. Numer. Anal., 4 (1967), pp. 89-102.

[7] M. Griebel, *Parallel multigrid methods on sparse grids*, in Multigrid Methods, III, Proceedings of the 3rd European Conference on Multigrid Methods, Bonn, 1990, W. Hackbusch and U. Trottenberg, eds., Internat. Ser. Numer. Math., 98 (1991), Birkhäuser, Basel, pp. 211-221, 1991; also Technische Universität München, SFB-Bericht Nr. 342/30/90 A.

[8] ———, *A parallelizable and vectorizable multi-level algorithm on sparse grids*, in Parallel Algorithms for Partial Differential Equations: Proceedings of the Sixth GAMM-Seminar, Kiel, 19.1.-21.1. 1990, Notes on Numerical Fluid Mechanics (Vol. 31), W. Hackbusch, ed., Vieweg, Braunschweig, pp. 94-100, 1991; Technische Universität München, SFB-Bericht Nr. 342/20/90 A.

[9] M. Griebel and P. Oswald, *On additive Schwarz preconditioners for sparse grid discretizations*, Friedrich-Schiller-Universität Jena, Forschungsergebnisse der mathematischen Fakultät, Jena, 1992.

[10] W. Hackbusch, *Iterative Lösung großer schwachbesetzter Gleichungssysteme*, Teubner, Stuttgart, 1993.

[11] G. Peters and J. H. Wilkinson, *$Ax = \lambda Bx$ and the generalized eigenproblem*, SIAM J. Numer. Anal., 7 (1970), pp. 479-492.

[12] H. Yserentant, *On the multi-level splitting of finite element spaces*, Numer. Math. 49 (1986), pp. 379-412.

[13] C. Zenger, *Sparse grids*, in Parallel Algorithms for Partial Differential Equations: Proceedings of the Sixth GAMM-Seminar, Kiel, 19.1.-21.1. 1990, Notes on Numerical Fluid Mechanics (Vol. 31), W. Hackbusch, ed., Vieweg, Braunschweig, 1991; also Technische Universität München, SFB-Bericht Nr. 342/18/90 A.

[14] ———, *Hierarchische Datenstrukturen für glatte Funktionen mehrerer Veränderlicher*, in Informatik und Mathematik, M. Broy, ed., Springer, Berlin, pp. 142-150, 1991.

[15] ———, *Funktionale Programmierung paralleler Algorithmen für numerische und technisch-wissenschaftliche Anwendungen*, in GAMM-Seminar über Numerische Algorithmen auf Transputer-Systemen, Heidelberg, 31.5.-1.6. 1991, Teubner-Skripten zur Numerik, G. Bader, G. Wittum, and R. Rannacher, eds., Teubner, Stuttgart, 1992.

SIAM J. SCI. COMPUT.
Vol. 17, No. 3, pp. 647-657, May 1996

005

AN EFFICIENT ITERATIVE SOLUTION METHOD FOR THE CHEBYSHEV COLLOCATION OF ADVECTION-DOMINATED TRANSPORT PROBLEMS*

A. PINELLI†, W. COUZY‡, M.O. DEVILLE‡, AND C. BENOCCI†

Abstract. A new Chebyshev collocation algorithm is proposed for the iterative solution of advection-diffusion problems. The main features of the method lie in the original way in which a finite-difference preconditioner is built and in the fact that the solution is collocated on a set of nodes matching the standard Gauss–Lobatto–Chebyshev set only in the case of pure diffusion problems. The key point of the algorithm is the capability of the preconditioner to represent the high-frequency modes when dealing with advection-dominated problems. The basic idea is developed for a one-dimensional case and is extended to two-dimensional problems. A series of numerical experiments is carried out to demonstrate the efficiency of the algorithm. The proposed algorithm can also be used in the context of the incompressible Navier–Stokes equations.

Key words. advection-diffusion, collocation, Chebyshev, preconditioning, finite difference, staggered grid

AMS subject classifications. 65N35, 76D99

1. Introduction. The need to solve implicit equations is a basic requirement for spectral algorithms. For both steady problems, whose solution is sought through a time marching computation, and for unsteady calculations, spectral methods are often feasible only if an implicit or a semi-implicit procedure is introduced. A typical example is the solution of the unsteady Navier–Stokes equations. Many numerical solution algorithms for this problem consist of an implicit treatment of the diffusive terms in combination with an explicit scheme for the convective terms [1]. This approach results in a severe restriction to the admissible time step when the flow dynamics are dominated by convection. Alternatively, the (linearized) convective terms can be treated implicitly as well, yielding a large admissible time step. As an example of the latter family of methods for the Navier–Stokes equations on a domain Ω, we consider the following semidiscretized version of the projection method proposed by Shen [2].

$$\frac{1}{\kappa}\left(\hat{u}^{n+1}-u^{n}\right)-\nu\,\triangle\,\hat{u}^{n+1}+\left(u^{n}\cdot\nabla\right)\hat{u}^{n+1}=b^{n+1}, \tag{1}$$

$$\hat{u}^{n+1}_{\partial\Omega}=0, \tag{2}$$

$$\frac{1}{\kappa}\left(u^{n+1}-\hat{u}^{n+1}\right)+\nabla\phi^{n+1}=0, \tag{3}$$

$$\nabla\cdot u^{n+1}=0, \tag{4}$$

$$u^{n+1}\cdot\vec{n}=0, \tag{5}$$

where u is the velocity, ϕ is an approximation of the pressure, κ is the time step, $\vec{n}$ is the normal vector to the boundary $\partial\Omega$, b is a force vector, and the superscript n represents the time iteration index. Equation (1) is a series of scalar advection-diffusion equations, each one having as unknown a velocity component.

Here, we will focus on a particular algorithmical aspect related to equation (1), when the spatial operators are discretized by the Chebyshev collocation technique [1]. To this end we

*Received by the editors August 13, 1993; accepted for publication (in revised form) March 2, 1995. This text presents research results of the Belgian Incentive Program "Information Technology-Computer Science of the Future," initiated by the SPPS (Services du Premier Ministre, Programmation de la Politique Scientifique). All correspondence should be sent to M. Deville, Ecole Polytechnique Fédérale de Lausanne, DGM-IMHEF, 1015 Ecublens, Switzerland.

†Von Karman Institute for Fluid Dynamics, Chaussée de Waterloo 72, 1640 Rhode Saint Genèse, Belgium.

‡Université Catholique de Louvain, CESAME, Av. G. Lemaître 4-6, 1348 Louvain-la-Neuve, Belgium. Present address: Ecole Polytechnique Fédérale de Lausanne, DGM-IMHEF, 1015 Ecublens, Switzerland (deville@imhfhp28.epfl.ch).

will consider the following one-dimensional equation, meant to model the linearized equation (1):

$$-\epsilon \frac{d^2u}{dx^2} + p(x)\frac{du}{dx} = b(x) \qquad \text{on } \Omega = (-1,1), \tag{6}$$

$$u = 0 \qquad \text{on } \partial\Omega. \tag{7}$$

The two-dimensional formulation of (6–7) reads

$$-\epsilon \,\Delta\, u + p(x,y)\frac{\partial u}{\partial x} + q(x,y)\frac{\partial u}{\partial y} = b(x,y) \qquad \text{on } \Omega = (-1,1)^2, \tag{8}$$

$$u = 0 \qquad \text{on } \partial\Omega, \tag{9}$$

where the given functions $p(x)$, $p(x, y)$, and $q(x, y)$ can be considered as representatives of the velocity field at the previous time step.

The collocation procedure applied to (6)-(7) or (8)-(9) yields the linear system

$$\mathbf{L}U = F. \tag{10}$$

Our goal is the design of an efficient iterative procedure to solve problem (10), such that the convergence does not depend on the ratio between local advection and the positive diffusion coefficient ϵ. An effective solution technique does not only depend on the choice of the iterative scheme but also on the efficiency of the preconditioner. We propose a particular preconditioned collocation scheme that relies on the scheme introduced by Funaro [3] for the Legendre collocation method. This method is based on the collocation of the equations on a particular set of points (termed as "staggered grid") which is determined as a deviation from the original Gauss-Lobatto-Chebyshev (GLC) quadrature nodes in function of the local ratio between advection and diffusion. In this paper, we will extend the Funaro scheme to the Chebyshev collocation method, by introducing an original definition of the staggered grid that does not rely on the Sturm-Liouville problem associated with Legendre polynomials. We will show that such a grid exists and is unique. Moreover, the numerical procedure to compute the staggered grid is simplified, yielding important savings of CPU time.

2. One-dimensional model. In this paragraph we will give some heuristic considerations to highlight the difficulties that arise when one tries to precondition discrete advection-diffusion systems dominated by the advective terms. The same reasoning will introduce the basic ideas the proposed scheme relies on.

We start considering the one-dimensional case (6)-(7), with the given function $p(x)$ strictly positive on all the domain $\Omega = (-1, +1)$. The Chebyshev polynomial of order N is denoted by $T_N(x)$. The GLC points $\{x_i\}$ are defined as the zeros of $(x^2 - 1)T'_N(x)$ and are therefore given by

$$x_i = \cos\left(\frac{\pi i}{N}\right), \qquad i = 0, \ldots, N. \tag{11}$$

For future use, we introduce as well the set of corresponding midpoints $\{\xi_i\}$, defined as the zeros of $T_N(x)$:

$$\xi_i = \cos\left(\pi\frac{2i+1}{2N}\right), \qquad i = 0, \ldots, N-1. \tag{12}$$

Next we consider a polynomial $z(x) = T'_N(x)$ (sketched in Figure 1) vanishing in all internal GLC points, according to (11). Such a polynomial represents a high-frequency mode, typical of a spectral approximation, that standard finite-difference operators fail to represent for advection-dominated problems. To explain this, we remark that the centered finite-

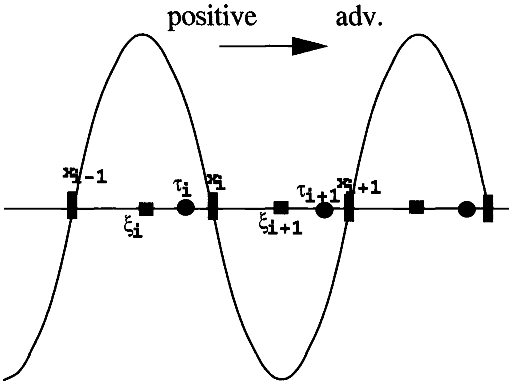


FIG. 1. *The polynomial* $z(x)$.

difference approximation of the second-order derivative in x_i (which is close to zero) is given by $\frac{h_{i+1}z(x_{i-1})-(h_i+h_{i+1})z(x_i)+h_iz(x_{i+1})}{0.5\,h_ih_{i+1}(h_i+h_{i+1})} = 0$, with $h_i = x_i - x_{i-1}$. An approximation of the first-order derivative (which attains its maximum value in x_i) will also yield zero, since $\frac{z(x_{i+1})-z(x_{i-1})}{h_i+h_{i+1}} = 0$. This observation explains the degrading behavior of classical finite-difference schemes constructed on the original GLC nodes for advection-dominated problems. In such a case, it is essential to provide a better approximation for the first-order derivative. To this end, we notice that the first-order derivative in the midpoint ξ_i is much more accurately approximated ($z'(\xi_i) \approx 0$) using the values of $z(x_i)$ and $z(x_{i-1})$. Such a heuristic approach is somehow confirmed by the theorem of Funaro [4] for first-order operators. For advection-diffusion problems, we are therefore induced to believe that the finite-difference preconditioner has to be evaluated on a special "staggered grid," although constructed using data at the GLC nodes. To set up this staggered grid, first we apply the operator (6) to the polynomial $z(x)$ that mimics the behavior of a high-frequency eigenfunction, and then we compute the zeros $\{\tau_i\}$ of the residual polynomial $f(x)$:

$$f(x) = -\epsilon z''(x) + p(x)z'(x). \tag{13}$$

For a purely second-order operator ($p(x) = 0$), the points $\{\tau_i\}$ coincide with the original set of GLC nodes, while for a first-order operator ($\epsilon = 0$) they drop on the GLC midpoints $\{\xi_i\}$. For a general advection-diffusion operator the following estimates hold:

$$\begin{aligned} \xi_i < \tau_i < x_i \text{ if } p(x) > 0, \\ x_i < \tau_i < \xi_{i+1} \text{ if } p(x) < 0. \end{aligned} \tag{14}$$

To implement the above theory, a polynomial $z(x)$ has to be found that mimics a high-frequency mode (i.e., $z(x_i) = 0$ and $z'(x_i)$ close to a local maximum or minimum at the interior collocation nodes x_i [3]). Because our target is the determination of a "staggered grid," which has the same number of nodes as the original Gauss-Lobatto grid, we require that the corresponding function $f(x)$ (see (13)) vanishes once and only once in each interval defined by $[\xi_i, x_i]$ when $p > 0$ and $[x_i, \xi_{i+1}]$ when $p < 0$. For the Legendre case, the procedure of determining a function $z(x)$ that respects all the aforementioned requirements is simplified by the possibility of reducing the complexity of the formulation by using the associated Sturm-Liouville problem [3]. In the Chebyshev case on the contrary, no simplification can be taken on the same side. Moreover, in analogy with the Legendre case, if we let the first-order derivative of the Nth Chebyshev mode play the role of the function $z(x)$, it can be shown that function $f(x)$ does not necessarily present a zero in the required interval $[\xi_i, x_i]$. For these reasons we prefer to weaken the condition of $z(x)$ vanishing in the set $\{x_i\}$. Therefore, like Funaro [3] for the Legendre case, instead of requiring that

$$z(x_i) = 0, \qquad z''(x_i) \approx 0, \qquad \text{and} \qquad \exists! \tau_i \in [\xi_i, x_i] : f(\tau_i) = 0, \tag{15}$$

we introduce a new strategy to determine the staggered grid by asking that

$$z(x_i) \approx 0, \qquad z''(x_i) = 0, \qquad \text{and} \qquad \exists! \tau_i \in [\xi_i, x_i] : f(\tau_i) = 0. \tag{16}$$

In this respect, a good candidate for $z(x)$ is

$$z(x) = \int T_N(x)dx. \tag{17}$$

It is clear that $z''(x_i) = 0$. Although $z(x)$ is not exactly zero in the set of $\{x_i\}$, we will prove that it differs from zero only by terms of order $1/N^2$ and we will also show the existence and uniqueness of a zero for the function $f(x)$ in the required interval $[\xi_i, x_i]$.

First we prove that $z(x)$ is "almost" zero in the set of $\{x_i\}$.

THEOREM 2.1. *For all* $x_i (i = 1, ..., N-1)$ *the polynomial* $z(x)$ *defined in* (17) *satisfies*

$$|z(x_i)| < \frac{1}{N^2-1}.$$

Proof. Because $T_N(x) = \frac{1}{2}\frac{1}{N+1}T'_{N+1}(x) - \frac{1}{2}\frac{1}{N-1}T'_{N-1}(x)$, one has

$$\begin{aligned} z(x_i) &= \int T_N(x)dx|_{x_i} \\ &= \frac{1}{2}\frac{1}{N+1}T_{N+1}(x_i) - \frac{1}{2}\frac{1}{N-1}T_{N-1}(x_i) \\ &= \frac{1}{2}\frac{1}{N+1}\cos\left(\frac{(N+1)\pi i}{N}\right) - \frac{1}{2}\frac{1}{N-1}\cos\left(\frac{(N-1)\pi i}{N}\right) \\ &= \frac{1}{2}\frac{1}{N+1}\left\{\cos(\pi i)\cos\left(\frac{\pi i}{N}\right) - \sin(\pi i)\sin\left(\frac{\pi i}{N}\right)\right\} \\ &\quad -\frac{1}{2}\frac{1}{N-1}\left\{\cos(\pi i)\cos\left(-\frac{\pi i}{N}\right) - \sin(\pi i)\sin\left(-\frac{\pi i}{N}\right)\right\} \\ &= \frac{1}{2}\frac{(-1)^i x_i}{N+1} - \frac{1}{2}\frac{(-1)^i x_i}{N-1} = \frac{(-1)^{i+1}x_i}{N^2-1}. \qquad \square \end{aligned} \tag{18}$$

Next, we show that the zeros of $f(x)$ can be used to build the finite-difference grid. More precisely, we have the following theorem.

THEOREM 2.2. *For all* $x_i (i = 1, \ldots, N), \epsilon > 0, p(x)$ *piecewise constant on* $[\xi_i, \xi_{i+1}]$, *the following properties hold.*

If $p(x) > 0$ *then there exists a unique* $\tau_i \in [\xi_i, x_i]$ *such that* $f(\tau_i) = 0$.

If $p(x) < 0$ *then there exists a unique* $\tau_i \in [x_i, \xi_{i+1}]$ *such that* $f(\tau_i) = 0$.

Proof. First, we assume that $p(x)$ is a positive constant in $[\xi_i, \xi_{i+1}]$. In this case, $f(x)$ is a polynomial of degree N. If we prove that there is at least one zero in each interval $[\xi_i, x_i]$, the total number of zeros is N and hence every zero is unique since N corresponds to the degree of the polynomial $f(x)$. This leaves us to show that $f(\xi_i) \cdot f(x_i) < 0$. We have

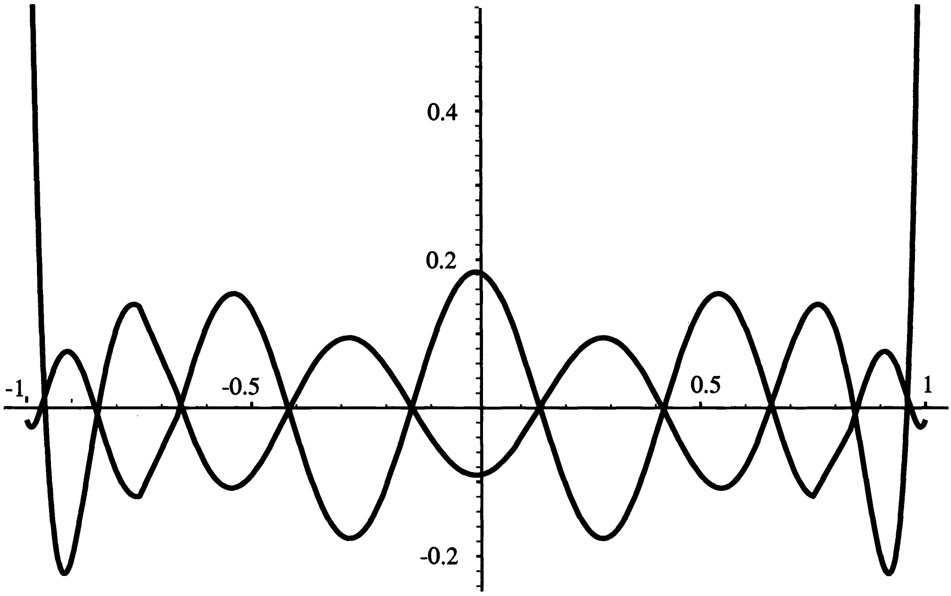


FIG. 2. $z(x)$ *and* $T_N'(x)/N^2$, *for* $N = 11$. $z(0) < 0$.

$f(\xi_i) = -\epsilon T_N'(\xi_i)$ and $f(x_i) = p(x_i)T_N(x_i)$. Because $p(x_i) > 0, \epsilon > 0$, it is enough to show that $T_N'(\xi_i) \cdot T_N(x_i) > 0$. This follows easily by noticing that

$$T_N'(\xi_i) = \frac{N}{\sqrt{1-\xi_i^2}} \sin(N \arccos(\xi_i)) = \begin{cases} < 0 & i \text{ odd}, \\ > 0 & i \text{ even}, \end{cases}$$

$$T_N(x_i) = \cos(N \arccos(x_i)) = \begin{cases} < 0 & i \text{ odd}, \\ > 0 & i \text{ even}. \end{cases} \tag{19}$$

In the same way, we demonstrate that $f(x)$ has exactly one zero in each interval $[x_i, \xi_{i+1}]$ if $p < 0$. To prove the theorem for $p(x)$ piecewise constant, we construct the family of polynomials $f_i(x) = -\epsilon z''(x) + p(x_i)z'(x)$. Each polynomial f_i has exactly one zero in the interval $[\xi_i, x_i]$ (if $p(x_i) > 0$) or in the interval $[x_i, \xi_{i+1}]$ (if $p(x_i) < 0$). Because $f(x)$ has the property that its restriction on $[\xi_i, \xi_{i+1}]$ coincides with the one of $f_i(x)$, it has exactly one zero in each desired interval. □

If $p(x)$ has a zero in an interval $[\xi_i, \xi_{i+1}]$, we map the finite-difference point on the Chebyshev node x_i.

In Figure 2, we display the function $z(x)$ and $T_N'(x)$. The latter function has $\{x_1, \dots, x_{N-1}\}$ as zeros and represents a high-frequency mode. We see that the zeros of the two polynomials practically coincide.

Based on the arguments presented in this section, we propose to use as a preconditioner the matrix associated with the centered finite-difference approximation to the problem (6)-(7) at the gridpoints $\{\tau_i\}$.

3. Two-dimensional formulation. Here, we consider the extension of the preconditioner to the two-dimensional problem (8)-(9). The GLC nodes are denoted by (x_i, y_j) and the midpoints by (ξ_i, η_j); cf. Figure 3. We propose to use the following function $z(x, y)$ that

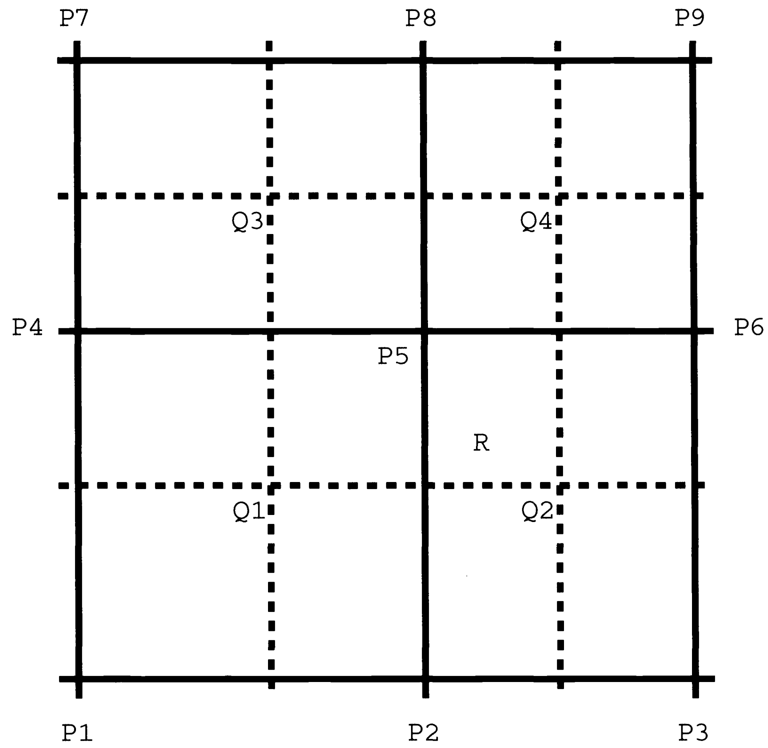


FIG. 3. *The position of the staggered point* $R = (\tau_{ij}, \upsilon_{ij})$ *in the GLC grid (straight lines) and the Gauss–Chebyshev grid (also called midpoints; dashed lines).* $P1 = (x_{i-1}, y_{j-1})$, $P5 = (x_i, y_j)$, $Q1 = (\xi_i, \eta_j)$, $Q4 = (\xi_{i+1}, \eta_{j+1})$, *etc.*

simulates a high-frequency mode in two dimensions:

$$z(x, y) = \int\int T_N(x) T_N(y) dx dy. \tag{20}$$

This function can be regarded as the tensor product of two one-dimensional functions $z(x)$ and $z(y)$. We find for all $x, y \in \Omega$ and for all $i, j = 0, \ldots, N$ that

$$z(x_i, y) \approx 0, \qquad z_{xx}(x_i, y) = 0, \qquad z(x, y_j) \approx 0, \quad \text{and} \quad z_{yy}(x, y_j) = 0. \tag{21}$$

Substitution of $z(x, y)$ in the two-dimensional analogue of (13) gives

$$f(x, y) = -\epsilon \Delta z(x, y) + p(x, y) z_x(x, y) + q(x, y) z_y(x, y). \tag{22}$$

The zeros $\{(\tau_{ij}, \upsilon_{ij})\}$ of the function $f(x, y)$ build the finite-difference grid for the preconditioner. In the following, we will assume p and q to be piecewise constant functions. Furthermore, we will focus on a rectangular domain Ω_{ij},

$$\Omega_{ij} = [\xi_i, \xi_{i+1}] \times [\eta_j, \eta_{j+1}], \tag{23}$$

on which $p(x, y) = p_{ij} > 0$ and $q(x, y) = q_{ij} > 0$. The zeros of $f(x, y)$ are given by the following relation:

$$\begin{aligned} & f(x, y) = 0 \qquad \Longleftrightarrow \\ & \{-\epsilon T_N'(x) + p_{ij} T_N(x)\} \int T_N(y) dy + \{-\epsilon T_N'(y) + q_{ij} T_N(y)\} \int T_N(x) dx = 0. \end{aligned} \tag{24}$$

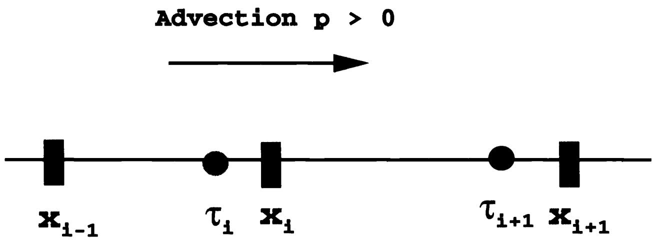


FIG. 4. *The position of three consecutive GLC nodes x_{i-1}, x_i, x_{i+1} on which the second-order Lagrange polynomial is constructed. This polynomial is evaluated in τ_i.*

From Theorem 2.1, we find

$$\exists!\tau_{ij} \in [\xi_i, x_i] : -\epsilon T_N'(\tau_{ij}) + p_{ij} T_N(\tau_{ij}) = 0, \tag{25}$$

$$\exists!\upsilon_{ij} \in [\eta_i, y_i] : -\epsilon T_N'(\upsilon_{ij}) + q_{ij} T_N(\upsilon_{ij}) = 0. \tag{26}$$

Hence, $\{(\tau_{ij}, \upsilon_{ij})\}$ are zeros of $f(x, y)$. Note that $f(x, y)$ is a polynomial of order $(N + 1)^2$, which implies that not all the zeros are found by solving (25)-(26). However, this is not a serious problem, since these additional zeros are not used to build up the finite-difference grid. If $p(x, y)$ assumes the value zero at one point at least in the rectangle Ω_{ij}, we will take x_i as the x-coordinate of the finite-difference node (analogously for $q(x, y)$).

4. Algorithmic aspects. We start focusing upon the one-dimensional model equation (6)-(7). The first step consists of determining the zeros of polynomial $f(x)$ (13). The computational cost to solve such a problem should be kept as low as possible. This issue is especially important when the preconditioner is used for nonlinear problems, like the Burgers or Navier-Stokes equations, in which the finite-difference grid has to be computed at each iteration. Therefore, we propose not to solve (13) by an expensive Newton method, as has been proposed by Funaro in [3]. Instead, we replace $f(x)$ by a piecewise parabolic function $d_i(x)$ on each $[\xi_i, x_i]$ (when $p > 0$) or on each $[x_i, \xi_{i+1}]$ (when $p < 0$). The zeros of these parabolas can be found by a simple and cheap formula. In the case of $p > 0$, we define the parabola $d_i(x)$ on the interval $[\xi_i, x_i]$ by the following three conditions.

$$d_i(\xi_i) = f(\xi_i), \qquad d_i(x_i) = f(x_i), \qquad \text{and} \qquad (d_i)_x(\xi_i) = f_x(\xi_i). \tag{27}$$

In this way, the second-order polynomial d_i has exactly one zero in $[\xi_i, x_i]$. Because our preconditioner is designed for advection-dominated problems, the first-order derivative is matched in ξ_i and not in x_i (27), yielding a better approximation for the zero which is, in this case, close to ξ_i.

Once the zeros $\{\tau_i\}$ (i.e., the "staggered grid") are found, for any i we approximate equation (6) with second-order Lagrange polynomials based on the data from three successive Chebyshev nodes x_{i-1}, x_i, x_{i+1} and collocated on the corresponding staggered point τ_i (see Figure 4). The general algorithm for the one-dimensional equation (6) with homogeneous Dirichlet boundary conditions (7) reads as follows.

- Select a guess solution U^0.
- Compute the zeros $\{\tau_i\}$ of the polynomial $f(x)$ defined in (13).
- Construct the finite-difference preconditioner $\mathbf{H}$ as

$$\mathbf{H} = \begin{bmatrix} b & c & 0 & & & & & \\ a & \ddots & \ddots & \ddots & & & & \\ & \ddots & \ddots & \ddots & \ddots & & & \\ & 0 & a_i & b_i & c_i & 0 & & \\ & & & \ddots & \ddots & \ddots & & \\ & & & & \ddots & \ddots & \ddots & \\ & & & & & \ddots & \ddots & \ddots \\ & & & & & & \ddots & \ddots \end{bmatrix} \tag{28}$$

with the following entries:

$$\begin{aligned} a_i &= p(\tau_i)l'_{i-1}(\tau_i) - \epsilon l''_{i-1}(\tau_i), \\ b_i &= p(\tau_i)l'_i(\tau_i) - \epsilon l''_i(\tau_i), \\ c_i &= p(\tau_i)l'_{i+1}(\tau_i) - \epsilon l''_{i+1}(\tau_i), \end{aligned} \tag{29}$$

as

$$l_i(x) = \frac{(x - x_{i-1})}{(x_i - x_{i-1})} \cdot \frac{(x - x_{i+1})}{(x_i - x_{i+1})}. \tag{30}$$

- Instead of $\mathbf{L}U = F$ solve

$$\mathbf{H}^{-1}\mathbf{L}U = \mathbf{H}^{-1}F \tag{31}$$

by an iterative algorithm. We remind that the system $\mathbf{L}U = F$ (see (10)) is the algebraic formulation of the spectral collocation method at the Chebyshev nodes $\{x_i\}$. Let us consider a simple preconditioned Richardson algorithm [1].

$$\begin{cases} \mathbf{H}\delta U = \quad R\tilde{H}S, \\ U^{n+1} = U^n + \delta U, \end{cases} \tag{32}$$

with

$$RHS = \omega\left(F - \mathbf{L}U^n\right) = \mathbf{W}^{-1}R\tilde{H}S, \tag{33}$$

$\mathbf{W}$ is a mapping operator that interpolates the residual computed on the collocation grid $\{x_i\}$ to the staggered grid $\{\xi_i\}$ and ω a relaxation factor. At each step a finite-difference problem is solved by inverting a tridiagonal matrix with a right-hand side that is the "spectral" residual at the previous step interpolated on the staggered grid.

- Iterate until convergence.

The previously described procedure has been extended to the two-dimensional advection-diffusion equation (8)–(9). Here, the staggered grid is given by the set of points $\{(\tau_{ij}, \eta_{ij})\}$ that are always determined as a deviation from the original Gauss-Lobatto grid in terms of the ratio between local advection $(p(x, y), q(x, y))$, and the diffusion coefficient ϵ. In particular, according to relationships (23) and (24), we approximate $f_1(x) := -\epsilon T'_N(x) + p_{ij}T_N(x)$ and $f_2(y) := -\epsilon T'_N(y) + q_{ij}T_N(y)$ locally by two one-dimensional parabolas. The zeros of these parabolas can be found following the procedure previously described for the one-dimensional case.

TABLE 1

Eigenvalues of the matrix $\mathbf{H}^{-1}\mathbf{L}$ *for different values of* ϵ.

Testcase	max $\Re(\lambda)$	min $\Re(\lambda)$	max $\Im(\lambda)$
$\epsilon = 10^{-2}$ 21 nodes	1.09	0.65	0.56
$\epsilon = 10^{-4}$ 21 nodes	2.31	0.65	0.15
$\epsilon = 10^{-5}$ 21 nodes	2.31	0.32	0.09
$\epsilon = 10^{-5}$ 41 nodes	2.31	0.32	0.10

The mapping operator for the two-dimensional case **W** is now chosen to be the local Taylor series expansion of the residual $r(x, y)$ of the current iteration:

$$r(\tau_{ij}, \eta_{ij}) \approx r(x_i, y_j) + (\tau_{ij} - x_i)\frac{\partial r}{\partial x} + (\eta_{ij} - y_j)\frac{\partial r}{\partial y} + \frac{1}{2}(\tau_{ij} - x_i)^2\frac{\partial^2 r}{\partial x^2}$$

$$(34) \qquad + \frac{1}{2}(\eta_{ij} - y_j)^2\frac{\partial^2 r}{\partial y^2} + (\tau_{ij} - x_i)(\eta_{ij} - y_j)\left(\frac{\partial^2 r}{\partial x \partial y}\right),$$

where the derivatives of the residual $r(x_i, y_j)$ are "spectrally" computed at the GLC nodes.

The finite-difference preconditioner **H** is now based on a nine-points stencil and is always constructed using second-order Lagrange polynomials. The nine entries of the stencil corresponding to the point (x_i, y_j) are denoted by $h_{i,j}^{(1)}, \ldots, h_{i,j}^{(9)}$ and are determined by imposing

$$(35) \qquad \begin{aligned} & h_{i,j}^{(1)}u_{i-1,j-1} + h_{i,j}^{(2)}u_{i,j-1} + h_{i,j}^{(3)}u_{i+1,j-1} \\ & + h_{i,j}^{(4)}u_{i-1,j} + h_{i,j}^{(5)}u_{i,j} + h_{i,j}^{(6)}u_{i+1,j} \\ & + h_{i,j}^{(7)}u_{i-1,j+1} + h_{i,j}^{(8)}u_{i,j+1}h_{i,j}^{(9)}u_{i+1,j+1} \\ = & -\epsilon \,\Delta\, u + p(x, y)\frac{\partial u}{\partial x} + q(x, y)\frac{\partial u}{\partial y} \text{ at } (\tau_{ij}, \eta_{ij}), \end{aligned}$$

meaning that equation (8) should be exactly satisfied for each Lagrange polynomial $u(x, y)$ of degree two in each variable. Such a procedure leads to a preconditioner with a bandwidth of $2N + 5$.

The iterative algorithm that we used for the two-dimensional case is an $orthomin\{5\}$ (meaning that the maximum dimension of the Krylov space kept in memory is 5; in other words the algorithm is restarted every 5 iterations) [5] preconditioned with the above-described matrix **H**.

5. Numerical experiments. To test the effectiveness of the developed preconditioner, we computed the eigenvalues of matrix $\mathbf{H}^{-1}\mathbf{L}$ for the one-dimensional case. The results are reported in Table 1 and refer to different values of the diffusion coefficient ϵ with the advection function $p(x)$ frozen at unity ($p(x) = 1$). The features of the preconditioned eigenvalues spectrum are summarized by giving the maximum and the minimum real parts together with the imaginary part. For the most severe test case at $\epsilon = 10^{-5}$ we also give the spectrum with two different sets of nodes (i.e., respectively, $N = 21$ and $N = 41$). This last result is intended to show the independence of the eigenvalues spectrum of the number of nodes. We recall here that in the nonpreconditioned case the conditioning number scales like $\sim N^2$ for the advective part and like $\sim N^4$ for the diffusive one [6]. In Figure 5, the eigenvalues spectrum is also shown for the cases $\epsilon = 10^{-2}$ and $\epsilon = 10^{-5}$.

To check the robustness of the method for the two-dimensional case, several tests were performed using different advective functions $p(x, y)$ and $q(x, y)$. In Table 2, the efficiency of the preconditioner in terms of number of iterations required to achieve machine precision (i.e.,

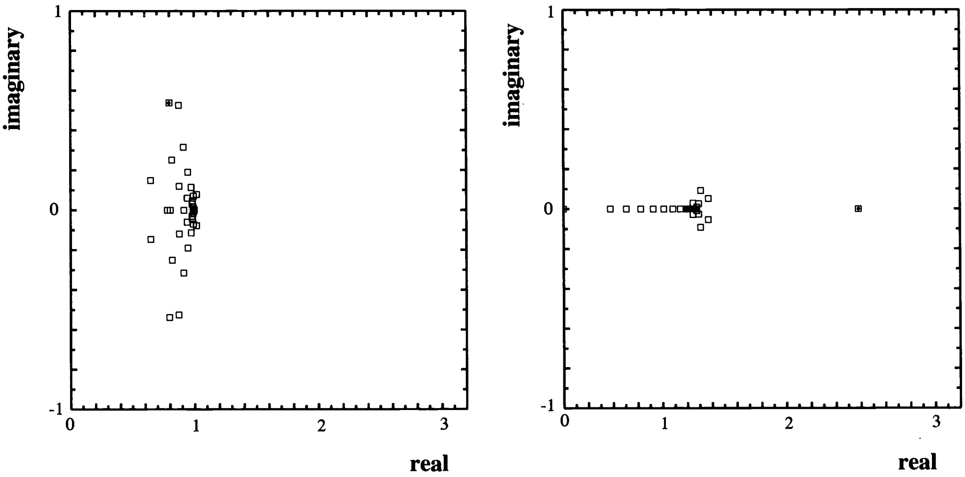


FIG. 5. *Preconditioned eigenvalues spectrum, $\epsilon = 10^{-2}$ and $\epsilon = 10^{-5}$, respectively.*

TABLE 2
Number of iterations to reduce the norm of the residual to 10^{-15}.

Testcase	Test 1	Test 2	Test 3	Test 4
Number of its.	65	45	56	82

L_∞ norm of the residuals at 10^{-15} for a double precision computation) is given at $\epsilon = 10^{-3}$ for, respectively,

$$\begin{array}{llll} & \textit{Test } 1. & p(x,y) = 1, & q(x,y) = 1, \\ & \textit{Test } 2. & p(x,y) = \sin(\pi x), & q(x,y) = 3x - y - 1, \\ (36) & \textit{Test } 3. & p(x,y) = 3x - y - 1, & q(x,y) = 3x^2 - y, \\ & \textit{Test } 4. & p(x,y) = \sin(\pi x)\sin(\pi y), & q(x,y) = \sin(\pi x)\sin(\pi y). \end{array}$$

As shown, the behavior is always satisfactory and the required number of iterations never exceeds by more than 30% the ones required for the constant advection case.

To test the developed preconditioner in a definite way, we considered equation (1) to be solved for the x-component of the velocity. The role of the advective functions $p(x, y)$ and $q(x, y)$ was played this time by an intermediate solution $u(x, y)$, $v(x, y)$ of a two-dimensional regularized driven cavity at a Reynolds number of 10000 (see Figure 6). The whole procedure converged to machine accuracy (10^{-14}) in less than 100 iterations, against the 27 iterations that are required when only the diffusive terms are treated implicitly (i.e., with a classical treatment of the diffusive terms with Crank-Nicolson and of the advective ones with Adams-Bashforth). Of course, a huge increase of the maximum allowed time step is expected from the fully implicit treatment and represents the payoff of the present method.

6. Conclusions. The basic target of the present work was the design of an efficient algorithm for the iterative solution of the advection-diffusion equations. To this end, a finite-difference preconditioner has been introduced on a "staggered" grid. The staggered points are given as the zeros of a polynomial that depends on the ratio between the local advection and the diffusion. Existence and uniqueness of these zeros is assured and the capacity of the preconditioner to represent high-frequency modes is shown. A cheap procedure to compute the staggered points has also been introduced. This is especially important when dealing with linearized problems, where the grid has to be constructed for every time step.

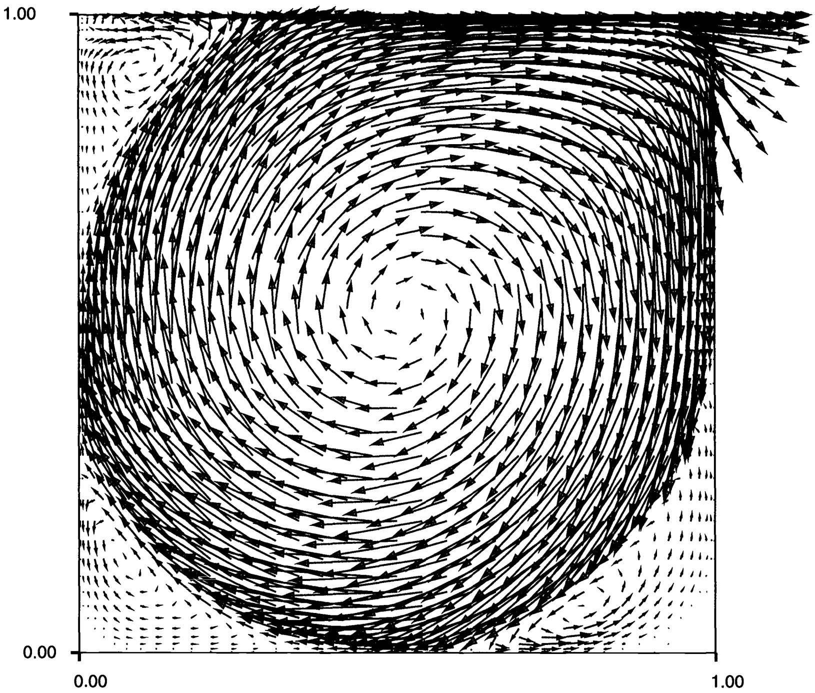


FIG. 6. *The chosen flow field.*

The efficiency of the implicit method is confirmed by numerical results for advection-dominated problems. Moreover, the distribution of the eigenvalues of the preconditioned operator seems almost independent of the value of the diffusion coefficient. The algorithm has also been applied to the projection method for the two-dimensional unsteady Navier–Stokes equations. Even for large Reynolds numbers, a very moderate number of iterations is needed to converge to machine accuracy.

Finally, we remark that we have been concerned with the development of an efficient preconditioning scheme and that no attempt has been made to cure problems related to numerical instabilities due to large Peclet numbers. Hence, we do not pretend that the presented method produces accurate results for the well-known boundary-layer problems for the values of ϵ and N that have been used to solve the analytical problem (36). Nevertheless, the present technique produces a condition number for the preconditioned operator that is almost independent of the number of nodes N. One way to take advantage of this feature is to increase N to be able to accurately capture boundary layers, without achieving prohibitive operation counts.

REFERENCES

[1] C. CANUTO, M. HUSSAINI, A. QUARTERONI, AND T. ZANG, Spectral Methods in Fluid Dynamics, Springer-Verlag, New York, 1988.

[2] J. SHEN, *On error estimates of projection methods for Navier–Stokes equation: First-order schemes*, SIAM J. Numer. Anal., 29 (1992), pp. 57–77.

[3] D. FUNARO, *A new scheme for the approximation of advection-diffusion equations by collocation*, SIAM J. Numer. Anal., 30 (1993), pp. 1664–1676.

[4] D. FUNARO AND E. ROTHMAN, *Preconditioning matrices for the pseudo-spectral approximation of first-order operators*, in Proceedings Finite Element Analysis in Fluids, T. J. Chung and G. R. Karr, eds., University of Alabama-Huntsville Press, Huntsville, AL, 1989.

[5] S. EISENSTAT, H. ELMAN, AND M. SHULTZ, *Variational iterative methods for non-symmetric systems of linear equations*, SIAM J. Numer. Anal., 20 (1983), p. 345.

[6] T. ZANG, C. STREETT, AND M. HUSSAINI, *Spectral Methods for Computational Fluid Dynamics*, Institute for Computer Applications in Science and Engineering, Report 89-13, Hampton, VA, 1989.

SIAM J. SCI. COMPUT.
Vol. 17, No. 3, pp. 658-665, May 1996

ON THE ORDER OF CONVERGENCE OF PRECONDITIONED NONLINEAR CONJUGATE GRADIENT METHODS*

M. AL-BAALI† AND R. FLETCHER‡

Abstract. An analysis is given of preconditioned nonlinear conjugate gradient methods in which the preconditioning matrix is the exact Hessian matrix at each iteration (or a nearby matrix). It is shown that the order of convergence of certain preconditioned methods is less than that of Newton's method when exact line searches are used, and an example is given.

Key words. unconstrained optimization, Newton's method, conjugate gradient methods, preconditioning

AMS subject classifications. 90C30, 65K05

1. Introduction.

Consider line search descent methods for solving the unconstrained optimization problem

$$\min_{x \in R^n} f(x), \tag{1}$$

where f is twice continuously differentiable. Each method generates a sequence of points $\{x^{(k)}\}$ for $k = 1, 2, \ldots$ until termination. The initial point $x^{(1)}$ is given. If $g^{(k)}[= \nabla f(x^{(k)})] = 0$, for some k, then the method terminates with $x^{(k)} = x^*$ (a stationary point). Otherwise, a search direction $s^{(k)}$ is defined for which

$$g^{(k)T} s^{(k)} < 0 \tag{2}$$

(the descent property). Then a new iterate is defined by the line search

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} s^{(k)}, \tag{3}$$

where $\alpha^{(k)}$ is a steplength chosen to minimize f along $s^{(k)}$ for which

$$g^{(k+1)T} s^{(k)} = 0. \tag{4}$$

In this case the line search is exact (e.g., Fletcher [2]).

The search direction in Newton's method is defined by

$$s^{(k)} = -G^{(k)-1} g^{(k)}, \tag{5}$$

where $G^{(k)}[= \nabla^2 f(x^{(k)})]$ is the Hessian matrix which, for $x^{(k)}$ sufficiently close to x^*, is usually positive definite.

We consider the case when a preconditioned version of a nonlinear conjugate gradient method is used to minimize f; the matrix W will denote the preconditioning matrix. At the first iteration, the search direction is defined by

$$s^{(1)} = -W^{(1)-1} g^{(1)}. \tag{6}$$

For $k > 1$,

$$s^{(k)} = -W^{(k)-1} g^{(k)} + \beta^{(k-1)} s^{(k-1)}, \tag{7}$$

*Received by the editors January 9, 1991; accepted for publication (in revised form) December 5, 1994.

†Department of Mathematics and Computer Science, United Arab Emirates University, United Arab Emirates (albaali@nyx.uaeu.ac.ae). This author is on leave from the Department of Mathematics, University of Damascus, Damascus, Syria.

‡Department of Mathematics and Computer Science, University of Dundee, Dundee, Scotland, UK.

where

$$\beta^{(k-1)} = \frac{\gamma^{(k-1)T} W^{(k)-1} g^{(k)}}{\gamma^{(k-1)T} s^{(k-1)}}, \tag{8}$$

and where

$$\gamma^{(k-1)} = g^{(k)} - g^{(k-1)}. \tag{9}$$

(This method is referred to as preconditioned conjugate gradient (PCG).) Resetting corresponds to defining $s^{(k)}$ as

$$s^{(k)} = -W^{(k)-1} g^{(k)}, \tag{10}$$

which in effect starts a new sequence of iterations. We note from (6) and (7), using (4), that the descent property (2) is satisfied for all k, since $W^{(k)}$ is chosen positive definite. (For further details, see Fletcher [2], for instance.)

The preconditioned Polak-Ribière (PPR) method is similar to the PCG method, except that $\beta^{(k-1)}$ is defined by

$$\beta^{(k-1)} = \frac{\gamma^{(k-1)T} W^{(k)-1} g^{(k)}}{g^{(k-1)T} W^{(k-1)-1} g^{(k-1)}} \tag{11}$$

rather than (8). But using (4), we observe from (6), (7), and (9) that

$$g^{(k-1)T} W^{(k-1)-1} g^{(k-1)} = -g^{(k-1)T} s^{(k-1)} = \gamma^{(k-1)T} s^{(k-1)},$$

for $k > 1$, which shows that the PCG and PPR methods are equivalent (i.e., (8) and (11) are identical) if exact line searches are performed. Thus the results in this paper are valid not only for the PCG method but also for the PPR method.

The preconditioned Fletcher-Reeves (PFR) method is also similar to the PCG method, except that formula (8) is replaced by

$$\beta^{(k-1)} = \frac{g^{(k)T} W^{(k)-1} g^{(k)}}{g^{(k-1)T} W^{(k-1)-1} g^{(k-1)}}. \tag{12}$$

For convenience, we also consider Newton-type methods which define $s^{(k)}$ by

$$s^{(k)} = -B^{(k)-1} g^{(k)}, \tag{13}$$

where $B^{(k)}$ is a symmetric positive definite matrix. This type of method is like Newton's method with line search, except that $G^{(k)}$ is approximated by $B^{(k)}$.

The focus of this paper is on the situation when a preconditioner is a close approximation of the exact Hessian. We show that in this case the above preconditioned methods are inferior in convergence rate to Newton's method (5). Since under certain conditions on f this method (with steps of unity) converges to x^* with R-order at least $p = \frac{1}{2}(1+\sqrt{5})$ and sometimes with Q-order at least 2 (e.g., Ortega and Rheinboldt [3], and Fletcher [2]), we approach our result as follows. In §2, we show, under those conditions on f and a strong one on $W^{(k)}$, that a PCG method with exact line searches converges to x^* with R-order at least p, and that bounds on the errors converge to zero with Q-order p. In §3, we illustrate the possibility that a PCG method and a PFR method converge with Q-order p and 1, respectively, by applying these methods to a simple quartic function. Because this result assumes that exact line searches are performed and that the exact Hessian matrix is used as the preconditioner, §3 also discusses certain numerical results which were obtained by using inexact line searches and $W^{(k)}$ is a close approximation of $G^{(k)}$. It is concluded, as for exact line searches, that the above preconditioned methods are less efficient than a Newton-type method defined by (13) and $B^{(k)} = W^{(k)}$.

This result has some theoretical significance, because even when a highly accurate Hessian approximation is used as the preconditioner and exact line searches are performed, the preconditioned methods converge slower than Newton's method or, for that matter, a Newton-type method having $B^{(k)} = W^{(k)}$.

2. Order of convergence. In order to obtain the results below, we begin by introducing some notation and by stating the assumptions we make about the objective function f and the preconditioning matrices.

Let $\|\cdot\|$ denote the usual Euclidean norm. We will use the notation $F(z) = O(z^r)$ as $z \to 0$, where r and z are scalars and F may be either a scalar, a vector, or a matrix. It means that there exists a positive constant c such that $\|F(z)\| \leq cz^r$ for z sufficiently close to zero (we do not bother to add the phrase "as $z \to 0$"). The error vector is defined by

$$h^{(k)} = x^{(k)} - x^* \tag{14}$$

and the difference between the current and new points is given by

$$\delta^{(k)} = \alpha^{(k)} s^{(k)} = x^{(k+1)} - x^{(k)}. \tag{15}$$

Assumptions 2.1. 1. The sequence of points $\{x^{(k)}\}$ remains in a closed, bounded, and convex level set $N = \{x : f(x) \leq f^{(1)}\}$, where $f^{(k)}$ denotes $f(x^{(k)})$, in which f is twice continuously differentiable.

2. The Hessian matrix G satisfies the Lipschitz condition

$$\|G(x) - G(x^*)\| \leq c_1 \|x - x^*\|, \tag{16}$$

where c_1 is a positive constant for all x in a neighborhood of x^*.

3. There exist positive constants c_2 and c_3 such that

$$c_2 \|z\|^2 \leq z^T G(x) z \leq c_3 \|z\|^2 \tag{17}$$

for all $z \in R^n$ and all $x \in N$.

4. The preconditioning matrices $W^{(k)}$ satisfy the equation

$$W^{(k)} = G^* + O(\|h^{(k)}\|) \tag{18}$$

for all $x^{(k)}$ sufficiently close to x^*.

Note that Assumption 2.1.3 and the convexity of N imply that f has a unique minimizer x^* in N. Note also that assumption (18) means that $W^{(k)}$ is a strongly consistent approximation of G^* (see Ortega and Rheinboldt [3] for instance). (For choice (18), the PCG and PFR methods are referred to as PCG1 and PFR1, respectively.)

Under the above assumptions, the Newton-type method (13) with steps of unity and $B^{(k)}$ defined by the right-hand side of (18) converges to x^* with Q-order at least 2 (e.g., Ortega and Rheinboldt [3]). However, this is not the case when the PCG1 and PFR1 methods are considered, even though both methods can be defined by (13) for some $B^{(k)-1}$. To illustrate this point, we state the following. It follows from (4), (7), (8), (9), (12), and (15) that the PCG and PFR methods define the search direction $s^{(k)}$ by (13) with $B^{(k)-1}$ replaced by

$$H_{PCG}^{(k)} = \left(I - \frac{\delta^{(k-1)}\gamma^{(k-1)T}}{\delta^{(k-1)T}\gamma^{(k-1)}}\right) W^{(k)-1}, \tag{19}$$

where I is the identity matrix, and

$$H^{(k)}_{PFR} = \left(I + \frac{\delta^{(k-1)}g^{(k)T}}{\delta^{(k-1)T}g^{(k-1)}}\right) W^{(k)-1}, \tag{20}$$

respectively. Since the matrix $H^{(k)}_{PCG}$ is singular, for all k, and, by (4), $H^{(k)-1}_{PFR} = W^{(k)}(I - \frac{\delta^{(k-1)}g^{(k)T}}{\delta^{(k-1)T}g^{(k-1)}})$ is not continuous at x^*, the inverse of both (19) and (20) cannot be defined by the right-hand side of (18). Thus we are not able to use a certain result of Ortega and Rheinboldt [3] and of Voigt [6] to obtain the rate of convergence for the PCG1 and PFR1 methods. Because the numerical results in §3 show that a PFR1 method converges linearly, we do not provide further details concerning this method. However, the following theorems are concerned with the PCG1 method. To obtain these results, we first present the following lemma.

LEMMA 2.1. *Let f satisfy Assumptions 2.1.1 and 2.1.3. Then for any minimization method that ensures that the value of f is reduced at every iteration, we have*

$$\delta^{(k)} = O(\|h^{(k)}\|) = O(\|g^{(k)}\|). \tag{21}$$

Proof. Using the mean value theorem and the fact that $g(x^*) = 0$, we obtain

$$f^{(k)} = f^* + \frac{1}{2}h^{(k)T}G(\xi)h^{(k)}, \tag{22}$$

where ξ lies on the line segment between x^* and $x^{(k)}$. Since N is a convex set and $x^{(k)} \in N$ (see Assumption 2.1.1), we have $\xi \in N$. Thus (22) and the uniform convexity assumption (17) imply

$$\frac{2}{c_3}(f^{(k)} - f^*) \le \|h^{(k)}\|^2 \le \frac{2}{c_2}(f^{(k)} - f^*). \tag{23}$$

Since $x^{(k+1)} \in N$, (23) is also satisfied with k replaced by $k+1$, which implies, using $f^{(k+1)} \le f^{(k)}$, that $\|h^{(k+1)}\|^2 \le 2(f^{(k)} - f^*)/c_2$. This inequality and the first one in (23) give

$$\|h^{(k+1)}\| \le c_4\|h^{(k)}\|, \tag{24}$$

where $c_4 = (c_3/c_2)^{1/2}$. From (14), (15), and (24) it follows that

$$\|\delta^{(k)}\| = \|h^{(k+1)} - h^{(k)}\| \le (c_4 + 1)\|h^{(k)}\|, \tag{25}$$

which shows that the first equality in (21) is correct. To prove the second equality in (21), use (25) and observe that the Taylor expansion theorem gives

$$g^{(k)} = G^*h^{(k)} + O(\|h^{(k)}\|^2), \tag{26}$$

which implies, using assumption (17), that

$$g^{(k)} = O(\|h^{(k)}\|) \tag{27}$$

and

$$h^{(k)} = O(\|g^{(k)}\|). \quad \square \tag{28}$$

THEOREM 2.2. *Let f satisfy assumptions 2.1.1-2.2.4 and assume that exact line searches are performed. Then the PCG1 method has the following properties:*

$$s^{(k)} = O(\|h^{(k)}\|), \tag{29}$$

$$\alpha^{(k)} = 1 + O(\|h^{(k-1)}\|). \tag{30}$$

Proof. In the following, we suppress the superscript "$^{(k)}$" on W, g, s, and α and replace the superscript "$^{(k-1)}$" by "$^{-}$". For convenience, we rearrange (7) with (8), using (19), as

$$s = -\left(I - \frac{\delta^- \gamma^{-T}}{\delta^{-T}\gamma^-}\right) W^{-1} g. \tag{31}$$

The Taylor expansion theorem shows that $g = g^- + G^\# \delta^-$, where $G^\# = \int_0^1 G(x^- + \delta^- t)dt$. Using the Lipschitz condition (16) and Lemma 2.1, it can be shown that $G^\# = G^* + O(\|h^-\|)$. Thus

$$g = g^- + G^*\delta^- + O(\|h^-\| \|\delta^-\|). \tag{32}$$

Substituting (18) and (32) in (31), we obtain, using assumption (17),

$$s = -\left(I - \frac{\delta^-\delta^{-T}G^*}{\delta^{-T}G^*\delta^-} + O(\|h^-\|)\right)\left(G^{*^{-1}} + O(\|h\|)\right) g.$$

Using (4), (24) (with k replaced by $k-1$ in both expressions), and (27), it follows that

$$s = -G^{*^{-1}} g + O(\|h\| \, \|h^-\|), \tag{33}$$

which, by (27), implies (29).

We now proceed to prove (30). From (4) and (32) with k replaced by $k+1$, we have

$$g^T s + \alpha s^T G^* s + O(\|h\| \|\delta\| \|s\|) = 0. \tag{34}$$

But from (33) it follows that $s^T G^* s = -g^T s + O(\|h\| \, \|h^-\| \, \|s\|)$. Substituting this expression in (34) and using the first equality in (21), (24), (28), and (29) it follows that

$$(1-\alpha) g^T s + O(\|g\|^2 \|h^-\|) = 0.$$

This result with (17), (28), and (33) imply that

$$\alpha = 1 + \frac{O(\|g\|^2\|h^-\|)}{g^T G^{*^{-1}} g + O(\|g\|^2\|h^-\|)} = 1 + O(\|h^-\|). \qquad \square$$

THEOREM 2.3. *Under the same assumptions of Theorem* 2.2, *the PCG*1 *method converges to x^* and has R-order at least $p = \frac{1}{2}(1+\sqrt{5})$. Moreover, the bounds on the errors exist and converge with exact Q and R order p.*

Proof. From (24), (26), and (33) we obtain

$$s^{(k)} = -h^{(k)} + O(\|h^{(k)}\| \, \|h^{(k-1)}\|).$$

Thus using (14), (15), (29), and (30) it follows that

$$h^{(k+1)} = O(\|h^{(k)}\| \, \|h^{(k-1)}\|)$$

or, equivalently,

$$\|h^{(k+1)}\| \leq c_5 \|h^{(k)}\| \, \|h^{(k-1)}\| \tag{35}$$

for some positive constant c_5. If we let $d_k = c_5\|h^{(k)}\|$, it follows from (35) that

$$d_{k+1} \leq d_k d_{k-1}. \tag{36}$$

If $x^{(k-1)}$ and $x^{(k)}$ are in a smaller neighborhood of x^* for which

$$d = \max(d_k, d_{k-1}) < 1, \tag{37}$$

then clearly $d_{k+1} \le d^2 < 1$. Thus $x^{(k+1)}$ is in the neighborhood and, by induction, the iterations are well defined for all k and $d_k \to 0$ (i.e., $h^{(k)} \to 0$). Thus the PCG1 method converges to x^*. Therefore, (36) implies that $\{d_k\}$ converges with R-order at least p, the positive root of the quadratic equation

$$t^2 - t - 1 = 0 \tag{38}$$

(e.g., Potra [4] and Voigt [6]).

To complete the proof, we proceed in the following way. We assume, without loss of generality, that $x^{(1)}$ is sufficiently close to x^* so that (36) holds, for all $k \ge 2$, and that (37) is true for $k = 2$. We let $\{b_k\}$, for $k \ge 1$, be a sequence of positive numbers defined by

$$b_k = \frac{1}{c_5} d^{a_k}, \tag{39}$$

where

$$a_k = \frac{1}{\sqrt{5}}(p^k - q^k), \tag{40}$$

and where $q = \frac{1}{2}(1 - \sqrt{5})$ is the negative root of (38). (Note that $b_k \to 0$, since $0 < d < 1$ and $p > 1$.) Since from (40) we obtain after a little manipulation that $a_1 = 1$, $a_2 = 1$, and $a_{k+1} = a_k + a_{k-1}$, for $k \ge 2$, it follows from (36) and (37) (by induction) that $c_5 b_k$ is an upper bound on d_k. Thus the sequence $\{b_k\}$ defines upper bounds on the errors $\|h^{(k)}\|$. To find its rate of convergence, we use (39) and (40) to obtain

$$\frac{b_{k+1}}{b_k^p} = c_5^{p-1} d^{a_{k+1} - p a_k} = c_5^{p-1} d^{q^k}. \tag{41}$$

Because $0 < d < 1$ and $-\frac{1}{2} < q^k < 1$, it follows from (41) that

$$c_6 b_k^p \le b_{k+1} \le c_7 b_k^p, \tag{42}$$

where $c_6 = c_5^{p-1} d$ and $c_7 = c_5^{p-1} d^{-1/2}$, which imply that $\{b_k\}$ converges with exact Q-order p, hence, with R-order p (e.g., Potra [4]). □

Although this result does not imply, in general, that the sequence $\{\|h^{(k)}\|\}$ converges with Q-order greater than one (e.g., Potra [4]), this sequence may have Q-order p in the following sense. If we assume that there exist constants for which the above bounds are attained, then we can expect that the sequence itself has this order. To illustrate this possibility, we present numerical results in the next section.

3. Numerical discussion. To illustrate the result in §2 that the PCG1 method (i.e., the PCG method with $W^{(k)}$ satisfying (18)) may converge with Q-order $p \approx 1.618$ and to show that the PFR1 method (i.e., PFR with (18) holds) may converge linearly, we consider the following experiments.

For the choice $W^{(k)} = G^{(k)}$, assuming an exact line search is performed at each iteration, we applied the PCG and PFR methods (referred to as PCG2 and PFR2, respectively) to the quartic function

$$f(x) = x_1^4 + x_1 x_2 + (1 + x_2)^2. \tag{43}$$

TABLE 1
Applications to problem (43).

	The PFR2 method		The PCG2 method		The Newton method	
k	$\|h^{(k)}\|$	$R1^{(k)}$	$\|h^{(k)}\|$	$R1.618^{(k)}$	$\|h^{(k)}\|$	$R2^{(k)}$
1	0.1119		0.1119		0.1119	
2	0.5362D-2	0.0479	0.5362D-2	0.1855	0.5362D-2	0.4282
3	0.1452D-3	0.0271	0.1592D-3	0.7513	0.8699D-5	0.3026
4	0.5924D-5	0.0408	0.4517D-6	0.6314	0.3151D-10	0.4164
5	0.1715D-6	0.0290	0.1958D-10	0.3615		
6	0.6496D-8	0.0379				
7	0.1983D-9	0.0305				

The starting point is given by $x^{(1)} = (0.75, -1.25)^T$ which is sufficiently close to the solution

$$x^* = (0.6958843861177639, -1.347942193058882)^T,$$

so that $G^{(k)}$, for all k, is positive definite (Fletcher [2]). (Note that the largest and smallest eigenvalues of G^* are approximately 6.06 and 1.75, respectively, and that Assumptions 2.1 hold.) The numerical results are given in columns 2, 3, 4 and 5 of Table 1. They were obtained in double precision with accuracy $2^{-56} \approx 0.14 \times 10^{-16}$. The run was stopped when $\|g^{(k)}\|^2 \leq 10^{-16}$. An examination of the ratios $R1.618^{(k)} = \|h^{(k)}\|/\|h^{(k-1)}\|^{1.618}$, given in column 5, shows that these ratios are approximately equal to a constant that illustrates that the PCG2 method has Q-order 1.618. Similarly an examination of the ratios $R1^{(k)} = \|h^{(k)}\|/\|h^{(k-1)}\|$, given in column 3, illustrates that the PFR2 method converges linearly. For convenience, we also applied Newton's method with exact line searches (defined by (3)-(5)) to the above problem. The results, given in columns 6 and 7 of Table 1, show, as expected, that the ratios $R2^{(k)} = \|h^{(k)}\|/\|h^{(k-1)}\|^2$ are nearly equal to a constant. It is clear that the PCG2 and PFR2 methods are inferior (the latter far worse) to the Newton method. (For the application of Newton's method with steps of unity, see Fletcher [2], who reported that the corresponding ratios $R2^{(k)}$ are close to 1.4.)

The above results show that even under the very strong condition (18) on the preconditioning matrix, the preconditioned methods, with exact line searches, converge slower than Newton's method. To see if these results generalize to inexact line searches and preconditioners that are only an approximation of the exact Hessian, these are conditions that are likely to hold in practice; it is worth mentioning an experiment of Al-Baali [1]. The author applied a PCG, PFR, and Newton-type method with inexact line searches to a set of standard nonlinear least squares problems. For this type of problem, the objective function is of the form $f(x) = r(x)^T r(x)$, where $r \in R^m$ and $m \geq n$. The choice

$$B^{(k)} = W^{(k)} = 2AA^T, \tag{44}$$

where $A = \nabla r(x^{(k)})^T$, the Gauss-Newton-Hessian of f, is a useful approximation of $G^{(k)}$ if $\|r(x^*)\|$ is sufficiently small (e.g., Fletcher [2]). Using (44), the three methods are referred to as PCG3, PFR3, and GN, respectively. (Note that the GN method is the so-called Gauss-Newton method and that the PFR3 method is essentially proposed by Ruhe [5].) The inexact line search algorithm used by Al-Baali [1] calculates a steplength $\alpha^{(k)}$ for which

$$|g^{(k+1)T}s^{(k)}| \leq -\sigma g^{(k)T}s^{(k)} \tag{45}$$

and

$$f^{(k+1)} \leq f^{(k)} + \rho\alpha^{(k)}g^{(k)T}s^{(k)}, \tag{46}$$

where $\rho \in (0, 0.5)$ and $\sigma \in (\rho, 1)$, the two-sided Wolfe–Powell conditions (e.g., Fletcher [2]). Note that in the limit $\sigma \rightarrow 0$, condition (45) reduces to (4). Using $\rho = 0.05$ and $\sigma = 0.1$ (which produces a fairly accurate line search), the latter three methods were implemented in single precision and terminated when

$$f^{(k)} - f^{(k+1)} \leq \epsilon \max(1, |f^{(k)}|), \tag{47}$$

where $\epsilon = 10^{-8}$. The reported numerical results (see [1] for details) show that for zero residual problems (in which $r(x^*) = 0$ and $G^* = 2A^*A^{*T}$), the performance of the GN method is better than that of both the PCG3 and PFR3 methods. Thus, in practice, the preconditioned methods converge slower than a Newton-type method having $W^{(k)}$ as a close approximation of $G^{(k)}$.

Therefore, if the preconditioning matrix $W^{(k)}$ is taken as a sufficiently close approximation of $G^{(k)}$, the Newton-type method with $B^{(k)} = W^{(k)}$ is preferable to the PCG and PFR methods.

Acknowledgments. The authors are indebted to Margaret Wright, Associate Editor, and to the two anonymous referees for many valuable comments and suggestions.

REFERENCES

[1] M. AL-BAALI, *Numerical experiments with the preconditioned conjugate gradient Gauss-Newton methods*, Damascus University Journal, 4 (1988), pp. 13–22.

[2] R. FLETCHER, *Practical Methods of Optimization*, Second Ed., Wiley, Chichester, England, 1987.

[3] J. M. ORTEGA AND W. C. RHEINBOLDT, *Iterative Solution of Nonlinear Equation in Several Variables*, Academic Press, New York, 1970.

[4] F. A. POTRA, *On Q-order and R-order of convergence*, JOTA, 63 (1989), pp. 415–431.

[5] A. RUHE, *Accelerated Gauss-Newton algorithms for nonlinear least squares problems*, BIT, 19 (1979), pp. 356–367.

[6] R. G. VOIGT, *Orders of convergence for iterative procedures*, SIAM J. Numer. Anal., 8 (1971), pp. 222–243.

SIAM J. SCI. COMPUT.
Vol. 17, No. 3, pp. 666-685, May 1996

007

PARALLEL SPARSE ORTHOGONAL FACTORIZATION ON DISTRIBUTED-MEMORY MULTIPROCESSORS*

CHUNGUANG SUN†

Abstract. In this paper, we propose a new parallel multifrontal algorithm for the orthogonal factorization of large sparse matrices on distributed-memory multiprocessors. We explore the use of block partitioning schemes in parallel sparse orthogonal factorization. Our block-oriented parallel algorithm for sparse orthogonal factorization achieves high performance by incurring strictly less communication overhead than the conventional nonblock algorithm, maintaining relatively balanced load distribution among processors, and accelerating the parallel numerical kernel via increased cache utilization. We analyze the performance of our parallel algorithm and present its arithmetic and communication complexities for regular grid problems. We report the experimental results of an implementation of our parallel algorithm on an Intel iPSC/860 machine. Through our theoretical analysis and experimental results, we demonstrate that our new block-oriented algorithm outperforms the conventional nonblock algorithm impressively.

Key words. parallel algorithms, sparse matrix, orthogonal factorization, multifrontal method, block partitioning scheme, distributed-memory multiprocessors

AMS subject classifications. 65F05, 65F50

1. Introduction.

Let A be a sparse $m \times n$ matrix with full column rank. The QR factorization of A is expressed as

$$A = Q \begin{bmatrix} R \\ 0 \end{bmatrix},$$

where Q is an $m \times m$ orthogonal matrix and R is an $n \times n$ upper triangular matrix. Usually Q is not formed explicitly.

A row merging scheme and a general row merging scheme for sparse QR factorization are proposed in [8] and [19], respectively. Sequential algorithms for sparse QR factorization are also considered in [11, 17, 22, 23]. Parallel algorithms are described in [5, 24, 27] for implementing the numeric phase of the general row merging scheme [19] on (multiple instruction multiple data) MIMD distributed-memory multiprocessors. A parallel algorithm is discussed in [16] for implementing the numeric phase of the row merging scheme [8] on (single instruction multiple data) SIMD architecture. Previously proposed parallel sparse QR factorization algorithms [5, 16, 24, 27] are based on the nonblock or single-row partition of the dense matrices involved in the numerical computation. The block-oriented approach to parallel sparse QR factorization has not been explored before.

In this paper we consider a block-oriented approach to parallel sparse QR factorization and describe a new parallel multifrontal algorithm for sparse QR factorization on distributed-memory multiprocessors. We propose block schemes for partitioning the upper trapezoidal matrices involved in the numerical computation. We explore a spectrum of block partitioning schemes ranging from the conventional single-row partition to the block partition with maximal block size. Our block-oriented approach results in an efficient parallel multifrontal algorithm for sparse QR factorization with low communication costs, relatively balanced load distribution among processors, and a high performance parallel numerical kernel due to the increased cache utilization.

*Received by the editors December 21, 1993; accepted for publication (in revised form) January 23, 1995. This work was supported in part by the Cornell Theory Center which receives funding from members of its Corporate Research Institute, the National Science Foundation, the Advanced Research Projects Agency, the National Institutes of Health, New York State, and IBM Corporation.

†Advanced Computing Research Institute, Cornell Theory Center, Cornell University, Ithaca, NY 14853-3801 (csun@cs.cornell.edu).

1. Find a column ordering P_1 for A such that $(AP_1)^T(AP_1)$ has a sparse Cholesky factor.
2. Compute the elimination tree of $(AP_1)^T(AP_1)$ from $G(A^TA)$ and P_1. Number the nodes of the elimination tree in postorder. Let the postorder be P_2 and set $P = P_1P_2$.
3. Determine the symbolic structure of R from $G(A^TA)$ and P, where $(AP)^T(AP) = R^TR$.
4. Compute the supernodal elimination tree of R from the elimination tree and the symbolic structure of R.
5. Perform numerical factorization by processing the supernodes in a topological ordering.

FIG. 1. *Sparse QR factorization.*

Both A and R are distributed to processors by rows. The supernodal structure of R is fully exploited. The performance of our parallel algorithm with a special block partition applied to regular grid problems is analyzed, and the arithmetic and communication complexity results of our analysis are presented. Experimental results of an implementation of our parallel algorithm on a 32-node Intel iPSC/860 machine are obtained for a collection of sparse matrix problems including regular grid problems and a general sparse problem with irregular sparsity structure. Our block-oriented approach to parallel sparse QR factorization achieves significant improvement in performance over the conventional nonblock approach described in [5].

In §2, a multifrontal sparse QR factorization scheme is reviewed. A new parallel multifrontal algorithm for sparse QR factorization is proposed in §3. The complexity analysis of our algorithm for regular grid problems is presented in §4. Experimental results are discussed in §5. Finally, concluding remarks are contained in §6.

2. A multifrontal sparse QR factorization algorithm. We briefly review a multifrontal approach to sparse QR factorization in this section. We refer the reader to [11, 17, 19] for further details. Let $A^TA = LL^T$, where L is the Cholesky factor of A^TA. It is well known that R^T is equal to L, apart from possible sign differences in the rows. Therefore, the elimination tree of A^TA can be defined in terms of the sparsity structure of R. Specifically, j is the parent of i if $r_{ij} (i < j)$ is the leading off-diagonal nonzero in the ith row of R. For simplicity, it is assumed that A^TA is irreducible and the corresponding elimination tree is indeed a tree. A node in the elimination tree corresponds to a row in R. Let $G(A^TA)$ denote the adjacency graph of A^TA. The steps involved in sparse QR factorization are described in Fig. 1.

Step 1 is done by applying a symmetric ordering such as the nested dissection ordering [7] or the minimum degree ordering [12] to $G(A^TA)$. Step 2 can be accomplished by using an algorithm described in [18]. The symbolic factorization step is discussed in [9]. There are several slightly different definitions of supernode [1, 2, 3, 10, 20]. Any of those definitions can be used in our approach. In an implementation of our algorithm, the fundamental supernode [2] is used. Let $\mathcal{R}_j$ denote the structure of row j of R, i.e., the set of nonzero column indices in row j of R. A fundamental supernode is a maximal set of contiguous rows $\{i, i+1, \dots, k\}$ in R such that $\mathcal{R}_j = \{j\} \cup \mathcal{R}_{j+1}$ for $i \le j < k$ and j is the only child of $j+1$ in the elimination tree which has been postordered. The first four steps are referred to as the *symbolic phase* of the sparse QR factorization and step 5 is referred to as the *numeric phase* of the sparse QR factorization.

A supernode K is associated with an upper trapezoidal matrix F_K called the *frontal matrix* of K. The sparsity structure of F_K is given by $\mathcal{R}_i$, where i is the least-numbered row in K. The number of columns in F_K is $|\mathcal{R}_i|$ and the number of rows in F_K is at most $|\mathcal{R}_i|$. Liu [19]

```
1  for each supernode $K$ in a topological ordering do
2      allocate space for $U_K$ and initialize $F_K$ to zero;
3      assemble $U_C$ into $F_K$ and discard $U_C$, where $C \in children(K)$;
4      for each $D \in children(K) \setminus \{C\}$ do
5          merge $U_D$ into $F_K$ and discard $U_D$;
6      for each $j \in K$ do
7          transform $A_j$ to an upper trapezoidal matrix $T_j$;
8          merge $T_j$ into $F_K$;
9      end for
10 end for
```

FIG. 2. *The numeric phase of a sequential multifrontal sparse QR factorization algorithm.*

has shown that F_K can be considered as a dense matrix for all practical purposes. Our parallel algorithm described in §3 works correctly regardless of whether F_K is upper trapezoidal or upper triangular. For simplicity, we assume that F_K is a dense upper triangular matrix. The first $|K|$ rows of F_K are rows $i, i+1, \ldots, i+|K|-1$ in R and they are called the *factor rows* of F_K. The upper triangular matrix obtained by removing the factor rows of F_K is called the *update matrix* of K and is denoted by U_K.

Let A_j denote the submatrix consisting of all rows of A whose first nonzeros are in column j of A. The number of nonzero columns in A_j is at most $|\mathcal{R}_j|$. Let *children*(K) be the set of children of K in the supernodal elimination tree. The numeric phase of a sequential multifrontal sparse QR factorization algorithm is described in Fig. 2. The storage for R is allocated before the numeric phase starts. The assembly of an update matrix U_C into F_K in line 3 can be done by a matrix *extend-add* operation introduced in [20], where C is any child of K. Merging U_D into F_K in line 5, reducing A_j to an upper trapezoidal matrix T_j in line 7, and merging T_j into F_K in line 8 can be accomplished by either Householder transformations or Givens rotations. In our implementation, U_D or T_j is merged into F_K by Givens rotations and A_j is reduced to T_j by Householder transformations. It is clear that this multifrontal approach mainly involves dense matrix computations. Furthermore, the update matrices can be handled by a stack data structure, so data locality is achieved. These desirable features are also present in our parallel algorithm.

3. A parallel multifrontal sparse QR factorization algorithm.

Several parallel algorithms have been described in [5, 24, 27] for implementing the numeric phase of sparse QR factorization on distributed-memory multiprocessors.

- In the algorithm discussed in [5], a frontal matrix F is mapped to a ring of processors by assigning the rows of F to the processors in a wrap-around manner. An update matrix U which needs to be merged into F is shifted among the ring of processors in a circular fashion. Before each shift the diagonal elements of U are annihilated against the frontal matrix F by Givens rotations and U becomes a smaller upper triangular matrix. This shift is continued until U is completely merged into F. This approach is referred to as the *row-oriented approach.*
- In the approach described in [24], a global row reduction algorithm is introduced to compute a frontal matrix. Parallel computation of the tree structure required for the multifrontal method is also discussed. In this approach, the amount of arithmetic work may increase as the number of processors increases.
- A parallel algorithm is discussed in [27] for implementing the sparse QR factorization scheme proposed in [11] in parallel. In this scheme, a frontal matrix is computed by row-oriented Householder transformations. Before the parallel computation of a frontal matrix is started, the relevant update matrices are redistributed among processors to ensure rela-

tively balanced load distribution. As expected, this redistribution improves load balancing by incurring additional communication overhead.

A drawback common to all of these algorithms is that frontal matrices are computed by communicating *single* rows; i.e., a message contains only one row or one transformed row of some frontal matrix. This can result in an excessive number of small messages communicated among processors. Because of the message start-up time, the overhead associated with a short message is proportionally higher than that associated with a long message.

To reduce communication costs, a block approach is suggested as a future research topic in [5]. A frontal matrix is partitioned into blocks with the same number of rows. If F is an $h \times h$ upper triangular matrix and is mapped to p processors, block size equal to h/p^2 is recommended in [5]. As shown in §5, the communication volume of this block approach is still relatively high. To the best of our knowledge, experimental results of the row-oriented approach and the block-oriented approach described in [5] have not been reported in the literature. In §5, performance results of these two approaches will be discussed.

3.1. An overall framework for parallel sparse QR factorization. We propose a new parallel algorithm for the numeric phase of the multifrontal sparse QR factorization described in §2. The symbolic phase of the sparse QR factorization is done sequentially. After the symbolic structure of R and the corresponding supernodal elimination tree are determined, the arithmetic work associated with each subtree is estimated. The supernodal elimination tree is mapped onto processors by a proportional mapping scheme described in [26]. The root of the supernodal elimination tree is partitioned among all processors. For a supernode that has already been mapped to a set of processors, each subtree rooted at a child of that supernode is allocated a subset of processors whose size is proportional to the workload associated with that subtree.

We consider the formation of a frontal matrix as a computational task. The supernodal elimination tree is the precedence graph among the computational tasks. Computation proceeds from leaves to root. Initially each processor is working on its own subtrees. Later on, processors cooperate to compute the frontal matrix of a partitioned supernode. When independent subtrees are being processed, no communication among processors is needed.

The overall parallel algorithm for the multifrontal sparse QR factorization is described in Fig. 3. The key operation in this algorithm is to merge an upper triangular matrix into another one in parallel as will be discussed in detail in the next subsection.

3.2. A parallel numerical kernel. Let V and W be two $h \times h$ upper triangular matrices. Consider merging V into W by Givens rotations on a set of p processors $\{\mu_0, \mu_1, \dots, \mu_{p-1}\}$, i.e., transforming the matrix $[\,{W \atop V}\,]$ into an upper triangular form by Givens rotations. Let V and W be identically partitioned into $k (k \le h)$ blocks by rows as

$$V = \begin{bmatrix} V_0 \\ V_1 \\ \vdots \\ V_{k-1} \end{bmatrix} \quad \text{and} \quad W = \begin{bmatrix} W_0 \\ W_1 \\ \vdots \\ W_{k-1} \end{bmatrix}.$$

Blocks V_i and W_i are assigned to processor μ_j, where $j = i \bmod p$. The block V_i is merged into blocks $W_i, W_{i+1}, \dots, W_{k-1}$, successively.

For a given number of processors, how the upper triangular matrices are partitioned determines the performance of merging V into W. Since there are p processors, the number of blocks should be at least p. Clearly, a block must contain at least one row. Therefore, $p \le k \le h$. The total number of possible block partitions for an $h \times h$ upper triangular matrix

```
for each supernode $K$ in a topological ordering do
    if $F_K$ is entirely mapped to $\mu$ then
        allocate space for $U_K$ and initialize $F_K$ to zero;
        assemble $U_C$ into $F_K$ and discard $U_C$, where $C \in children(K)$;
        for each $D \in children(K) \setminus \{C\}$ do
            merge $U_D$ into $F_K$ and discard $U_D$;
        for each $j \in K$ do
            merge rows in $A_j$ into $F_K$;
    else
        if $F_K$ is partially mapped to $\mu$ then
            allocate space for $\mu$'s portion of $U_K$;
            initialize $\mu$'s portion of $F_K$ to zero;
            assemble $U_C$ into $F_K$ in parallel and discard $\mu$'s portion of $U_C$,
                where $C \in children(K)$;
            for each $D \in children(K) \setminus \{C\}$ do
                merge $U_D$ into $F_K$ in parallel;
                discard $\mu$'s portion of $U_D$;
            end for
            merge relevant rows from $A$ into $F_K$ in parallel;
        end if
    end if
end for
```

FIG. 3. *A parallel multifrontal sparse orthogonal factorization algorithm on a processor μ.*

on p processors is

$$\sum_{k=p}^{h} \binom{h-1}{k-1} = \sum_{k=p}^{h} \frac{(h-1)!}{(k-1)!(h-k)!}.$$

For reasonable h and p, it is impractical to determine an optimal block partition by examining all possible block partitions. We consider two strategies for partitioning upper triangular matrices into blocks.

Equal-row partitioning scheme. First, we consider partitioning upper triangular matrices V and W into blocks with an equal number of rows. This scheme partitions an $h \times h$ matrix into $k = h/s$ blocks, where $s(1 \le s \le h/p)$ is the block size or the number of rows in a block. This scheme is referred to as the *equal-row partitioning scheme*. The special cases $s = 1$ and $s = h/p^2$ are considered in [5].

The number of blocks k determines the arithmetic work distribution of merging V into W. The larger the number of blocks, the more balanced the arithmetic work distribution. The worst and the best arithmetic work distributions are achieved for $k = 1$ and $k = h$, respectively. To maintain load balance, the number of blocks should be large; i.e., the block size should be small.

Let α be the start-up time for a message and β the transfer time per floating-point number. The total communication cost on all processors is

$$\sum_{i=1}^{k-1} i(\alpha + \beta(s(h - is) - (s-1)s/2)) \approx \frac{\alpha}{2}k^2 + \frac{\beta h^2}{6}k - \frac{\beta h^2}{4}.$$

Clearly, the communication cost is proportional to the number of blocks k. To keep the communication cost low, the number of blocks should be small; i.e., the block size should be large.

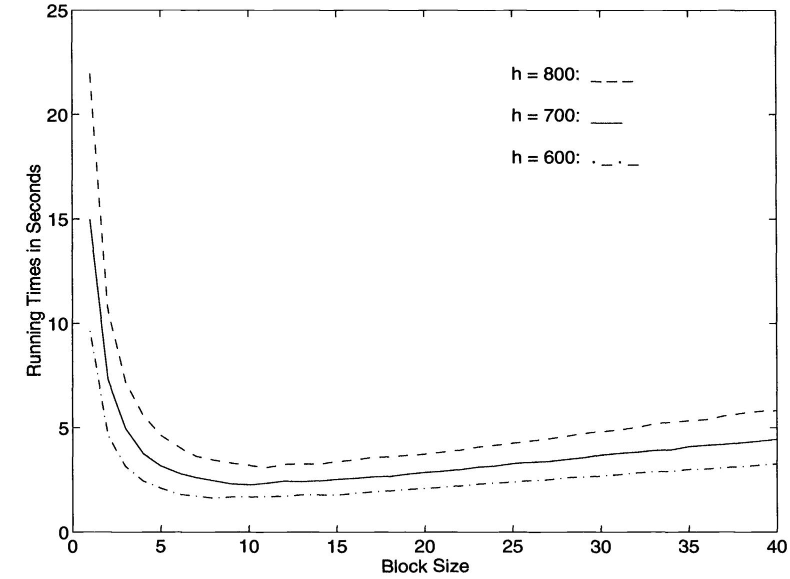


FIG. 4. *Equal-row partitioning scheme with $p = 32$ and $h = 600, 700, 800$ on an iPSC/860.*

An optimal block size minimizes the combined effect of the arithmetic work distribution and the communication cost; i.e., the overall execution time. To obtain an optimal block size or a block size that is close to optimal, it is necessary to examine how the algorithmic performance of merging V into W is dependent upon the block size.

Experiments are performed on an Intel iPSC/860 machine to investigate the relationship between the algorithmic performance and the block size used. Our results show that the special case $s = 1$ gives the worst performance. For fixed p and h, the running time initially decreases rapidly as the block size increases. Once the best performance is achieved, the running time increases slowly as the block size increases. The running times are virtually indistinguishable at the neighborhood of the optimal block size for given h and p. Therefore, it is not necessary to determine the optimal block size exactly. As an illustration, experimental results for $p = 32$ and $h = 600, 700, 800$ are shown in Fig. 4. Block sizes equal to $1, 2, \ldots, 40$ are used. The same kind of relationship is observed for other values of p and h.

The optimal block sizes for $p = 8$, 16, and 32 are shown in Table 1 under column "s." The columns under "time" and "mflops" show the corresponding running times and floating-point execution rates in megaflops per second, respectively. Floating-point operations are performed in double-precision arithmetic. Since the dense submatrices involved in the numerical kernel of our parallel sparse QR factorization algorithm are of order 100 to 900, the range of h shown in Table 1 is between 100 and 900. Let $s(h, p)$ denote an optimal block size for merging two $h \times h$ upper triangular matrices on p processors. Table 1 shows that optimal block sizes for $p = 8, 16, 32$ can be expressed as

$$(1) \qquad s(h, p) = \lceil 2h/(5p) \rceil.$$

Experimental results for $p = 2, 4$ have also been obtained. Optimal block sizes for $p = 2, 4$ can be expressed as

$$(2) \qquad s(h, p) = \lceil h/(3p) \rceil.$$

TABLE 1

Optimal block sizes and corresponding performance results on an Intel iPSC/860 machine.

h	$p=8$			$p=16$			$p=32$		
	s	time	mflops	s	time	mflops	s	time	mflops
100	5	0.057	18.875	4	0.048	22.414	2	0.053	20.299
200	10	0.220	37.735	6	0.177	46.902	4	0.155	53.559
300	16	0.546	50.691	8	0.423	65.432	5	0.353	78.407
400	20	1.106	58.954	10	0.839	77.716	6	0.668	97.610
500	25	1.929	65.775	13	1.292	98.204	7	1.123	112.982
600	30	3.074	71.147	15	2.079	105.197	8	1.615	135.421
700	35	4.545	76.277	18	3.061	113.257	9	2.300	150.731
800	40	6.505	79.448	20	4.329	119.382	11	3.090	167.251
900	44	8.996	81.712	22	5.835	125.978	12	4.189	175.479

Revised equal-volume partitioning scheme. Because of the triangular shape of the matrices, the equal-row partitioning scheme may result in load imbalance since the blocks become smaller and smaller toward the end. To correct this, we propose schemes that partition the upper triangular matrices V and W into blocks with nondecreasing number of rows. The *equal-volume partitioning scheme* introduced in [30] is one of such schemes. This scheme partitions V and W identically into k blocks by rows. Each block consists of a set of contiguous rows and contains about $h(h+1)/(2k)$ elements. Assume that the number of rows and the number of columns in the ith block are denoted by r_i and c_i, respectively. Then

$$\frac{c_i(c_i+1)}{2} \approx \frac{h(h+1)}{2}\left(\frac{k-i}{k}\right), \tag{3}$$

$$c_i \approx \sqrt{\frac{k-i}{k}}\,h, \quad \text{and} \quad r_i \approx \frac{\sqrt{k-i}-\sqrt{k-i-1}}{\sqrt{k}}\,h. \tag{4}$$

Notice that $\sum_{i=0}^{k-1} r_i = h$. It is easy to show $r_i \leq r_{i+1}$ for $0 \leq i < k-1$.

Now we propose a revised equal-volume partitioning scheme. Let k be the expected number of blocks to be partitioned. The revised equal-volume partitioning scheme actually partitions V and W into $l(l \leq k)$ blocks such that the ith block has $r_i + d$ $(d \geq 0)$ rows, where r_i is defined as in (4) and $d \geq 0$. This revised scheme is precisely the equal-volume partitioning scheme when $d = 0$. Clearly, $r_i + d \leq r_{i+1} + d$ for $0 \leq i < l-1$. The number of blocks l can be determined from $\sum_{i=0}^{l-1}(r_i + d) = h$. Our empirical results indicate that the optimal value for d is 3. Experimental results on 32 processors for $h = 800$ and $d = 0, 3, 6$ are shown in Fig. 5. In the remainder of this paper, $d = 3$ is assumed for the revised equal-volume partitioning scheme.

Depending on the number of processors used, the equal-row partitioning scheme with block size specified by expression (1) or (2) is referred to as the equal-row partitioning scheme with optimal block partition. Similarly, the revised equal-volume partitioning scheme with the expected number of blocks to be partitioned given by $h/s(h, p)$ is referred to as the revised equal-volume partitioning scheme with optimal block partition. The performance results of these two schemes with optimal block partition are compared in Fig. 6. The revised equal-volume partitioning scheme is, in general, more efficient than the equal-row partitioning scheme.

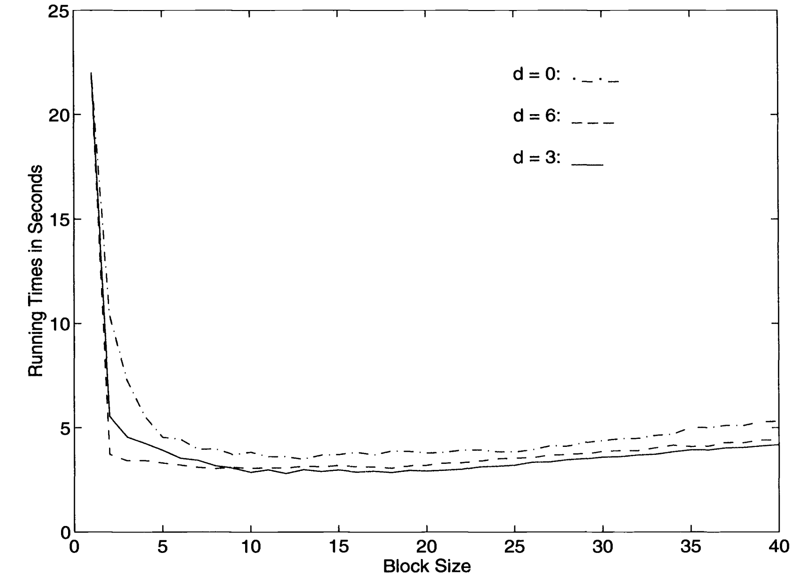


FIG. 5. *Revised equal-volume partitioning scheme with $p = 32$, $h = 800$, and $d = 0, 3, 6$ on an iPSC/860.*

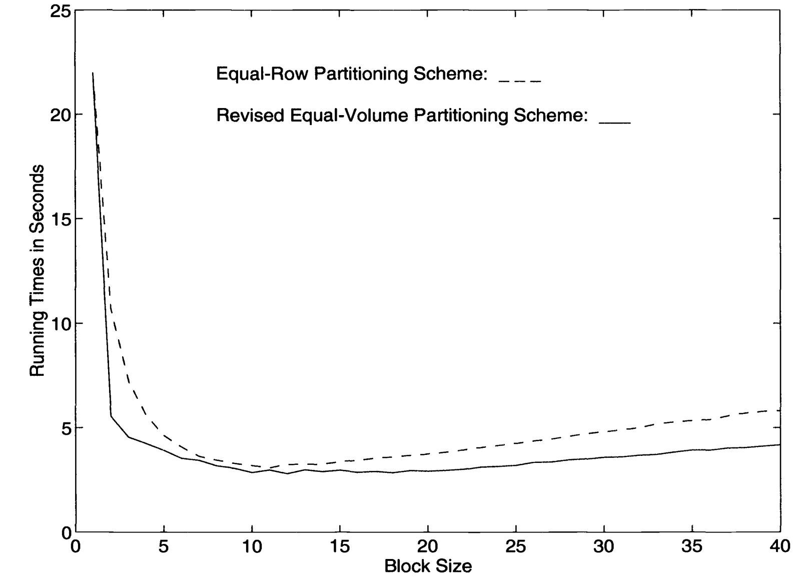


FIG. 6. *Comparison of the two partitioning schemes with $p = 32$ and $h = 800$ on an iPSC/860.*

Row-oriented kernel versus column-oriented kernel. Assume that V_i and W_j are 4×5 upper trapezoidal blocks. Two ways of merging V_i into W_j are shown below:

$$\begin{bmatrix} W_j \\ V_i \end{bmatrix} \Rightarrow \begin{bmatrix} x & x & x & x & x \\ & x & x & x & x \\ & & x & x & x \\ & & & x & x \\ 1 & 2 & 3 & 4 & x \\ & 5 & 6 & 7 & 8 \\ & & 9 & 10 & 11 \\ & & & 12 & 13 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} x & x & x & x & x \\ & x & x & x & x \\ & & x & x & x \\ & & & x & x \\ 1 & 2 & 4 & 7 & x \\ & 3 & 5 & 8 & 11 \\ & & 6 & 9 & 12 \\ & & & 10 & 13 \end{bmatrix}.$$

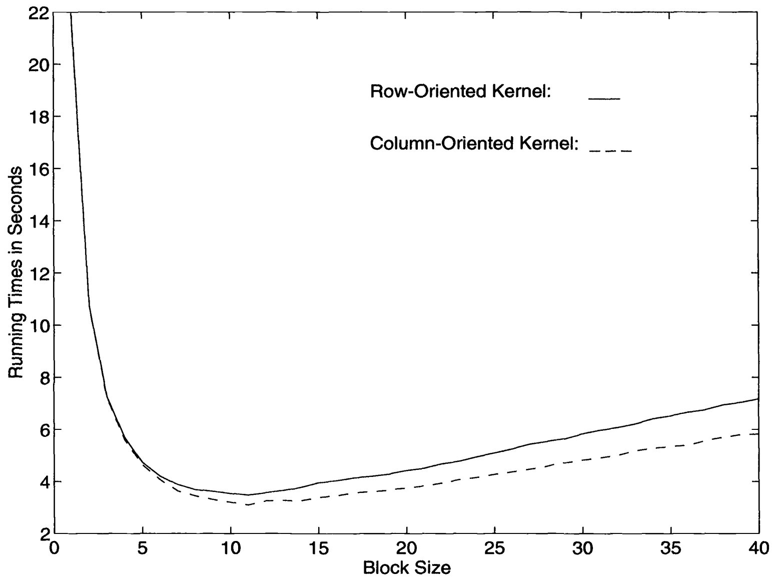


FIG. 7. *Comparison of the row-oriented and the column-oriented kernels with* $p = 32$ *and* $h = 800$ *on an iPSC/860.*

The numbers show the order in which the elements of V_i are annihilated. These two approaches correspond to merging V_i into W_j by rows and by columns, respectively. If the block size—i.e., the number of rows in V_i—is reasonably large, the column-oriented kernel is more efficient than the row-oriented kernel. The reason is that in the column-oriented kernel a rectangular submatrix of V_i is rotated into a row in W_j so this row in W_j is repetitively accessed. Therefore, the cache is better utilized and, consequently, the performance of the numerical kernel is improved. This is confirmed by the empirical results shown in Fig. 7. As the block size increases, a row in W_j is repetitively accessed more frequently in the column-oriented kernel. Therefore, the cache utilization of the column-oriented kernel gets better and it becomes increasingly more efficient than the row-oriented kernel as the block size increases.

3.3. Parallel computation of a frontal matrix. In order to compute a frontal matrix F_K in parallel, we have designed parallel algorithms for assembling the update matrix of a child of K into F_K, merging the update matrix of a child of K into F_K, and merging relevant rows from A into F_K. These algorithms correspond to lines 13, 16, and 19 in Fig. 3, respectively.

To simplify our discussion, we take the 7 x 7 nine-point grid with nested dissection ordering as an example. A portion of its supernodal elimination tree consists of supernodes $K = \{19, 20, 21\}$, $C = \{7, 8, 9\}$, and $D = \{16, 17, 18\}$. Supernodes C and D are the only children of supernode K. The frontal matrices F_K, F_C, and F_D are illustrated in Fig. 8, where the row numbers displayed with a frontal matrix show its sparsity structure.

We assume that F_K is partitioned among four processors $\{\mu_0, \mu_1, \mu_2, \mu_3\}$, F_C is partitioned among μ_0 and μ_1, and F_D is partitioned among μ_2 and μ_3, respectively. The solid lines inside the triangles in Fig. 8 show one way of partitioning F_K, F_C, and F_D into blocks.

Suppose that the frontal matrices F_C and F_D have already been computed. The update matrix U_C is the second block of F_C and is assigned to μ_1. Similarly, the update matrix U_D is the second block of F_D and is assigned to μ_3. Based on the sparsity structure of U_C, U_D, and F_K, the update matrices U_C and U_D are further partitioned into subblocks shown by the dash lines in Fig. 8.

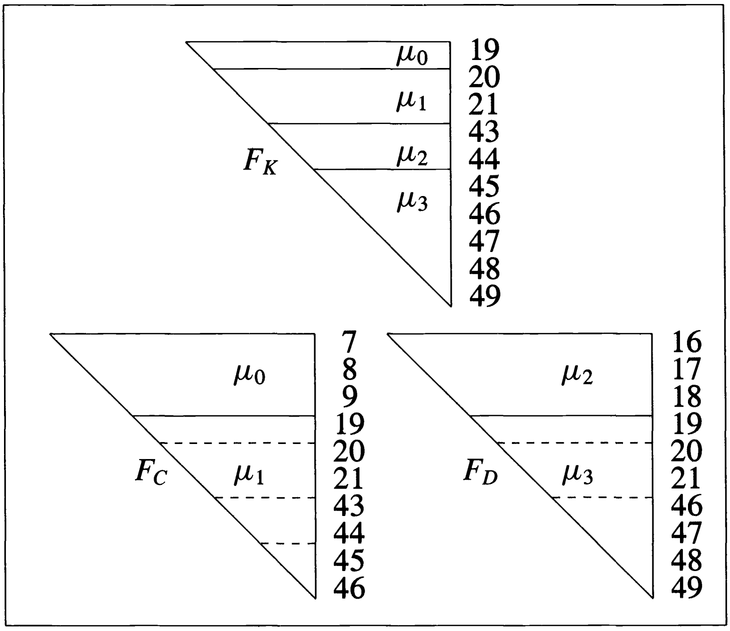


FIG. 8. *Parallel computation of a frontal matrix.*

Suppose U_C has already been assembled into F_K. Now we consider merging U_D into F_K. A subblock B of U_D is sent to a processor involved for computing F_K if the processor owns a block B_K of F_K such that the row numbers of B are contained in the row numbers of B_K. For instance, the second subblock of U_D owned by μ_3 is sent to μ_1 which owns the second block of F_K.

Once a subblock B is received by a processor, it is conceptually extended to a temporary block by the matrix *extend-add* operation introduced in [20] and this temporary block is then rotated into a block B_K owned by the processor. The result of rotating B into B_K is another upper trapezoidal block $\hat{B}$. If $\hat{B}$ is not empty, it is sent to the processor which owns the block just below the block B_K in F_K. Notice that the number of rows in $\hat{B}$ is less than or equal to the number of rows in B since block B_K may contain zero rows. Eventually, the newly created block $\hat{B}$ is merged into the remaining blocks of F_K, successively.

In order to complete the computation of F_K, a row from A whose first nonzero is in column j of A such that $i \leq j < i + |K|$ must be rotated into F_K, where i is the first row in K. The rows in the submatrix A_j, defined in §2, form a rectangular block with at most $|\mathcal{R}_j|$ nonzero columns. This block is rotated into F_K starting with a block B_K of F_K such that row $j - i$ of F_K is in block B_K. The resulting block is rotated into the remaining blocks of F_K, successively.

4. Analysis of regular grid problems. In this section, we present the complexity analysis of our parallel sparse QR factorization algorithm based on a special equal-volume partitioning scheme for regular grid problems. We compare our results with those provided in [5]. To be consistent with the results in [5], we count only the number of multiplications in our analysis.

We focus on the equal-volume partitioning scheme in which an $h \times h$ frontal matrix distributed among p processors is partitioned into p blocks. It seems to be very difficult or impossible to obtain the complexity results of an arbitrary equal-row or equal-volume partitioning scheme. The analytical results obtained for a special case of the equal-volume partitioning scheme would shed light on the performance of an arbitrary equal-row or equal-volume partitioning scheme from a theoretical angle.

Our goal is to obtain an upper bound on the overall execution time of our parallel sparse QR factorization algorithm. There are two levels of analysis. In the outer level, the supernodal elimination tree is mapped onto node processors by the subtree-to-subcube mapping scheme [13]. We identify a heaviest path of the supernodal elimination tree. The heaviest path consists of a sequence of computational tasks each of which is the computation of a frontal matrix. Careful exploration of the heaviest path leads to an upper bound on the execution time of our algorithm. In the inner level, we investigate the cost of computing a frontal matrix in parallel. The processors assigned for the task start execution at the same time. We identify a processor μ that has the most work to do and finishes its share of the task last. We characterize the sequence of subtasks executed by the processor μ. We show that there is no or very little gap between the subtasks executed by μ. Therefore, the time complexity of computing a frontal matrix in parallel is simply the sum of the time complexities of subtasks executed by μ.

The communication model assumed in our analysis is briefly described here. Consider the transmission of a message from one processor μ to another processor $\hat{\mu}$. The completion of the *send* operation on μ does not imply that the message has been received by $\hat{\mu}$, only that the message has been sent and the buffer space for the message on μ can be reused. Once the corresponding *receive* is issued by $\hat{\mu}$, processor $\hat{\mu}$ waits until the message arrives in the specified buffer. An example of this model is the communication model used on an Intel iPSC/2 or Intel iPSC/860 machine [15]. In this model, the cost of transmitting a message is charged to the receiver. Let τ denote the ratio of the time for transmitting one floating-point number from one processor to another processor to the time for one floating-point multiplication.

4.1. Analysis of the parallel numerical kernel. Consider two $h \times h$ upper triangular matrices V and W which are identically partitioned by our equal-volume partitioning scheme into p blocks as

$$V = \begin{bmatrix} V_0 \\ V_1 \\ \vdots \\ V_{p-1} \end{bmatrix} \quad \text{and} \quad W = \begin{bmatrix} W_0 \\ W_1 \\ \vdots \\ W_{p-1} \end{bmatrix}.$$

Blocks V_i and W_i are assigned to processor μ_i, $0 \le i < p$. The number of rows and the number of columns in the ith block are defined as in (4) in §3.2 except that k is replaced by p.

The total number of multiplications for merging V into W is $2h(h+1)(h+2)/3$. The tasks involved in merging V into W are illustrated in Fig. 9, where $M(V_i, W_i)(0 \le i < p)$ is the task for merging V_i into W_i, $M(V_i, V_j)(i < j)$ is the task for merging the transformed block V_i into the transformed block W_j, and $C(V_i, \mu_j, \mu_k)$ is the task for receiving the transformed block V_i from processor μ_j by processor μ_k, where $k = j + 1 \bmod p$. For simplicity, p is assumed to be 4 in Fig. 9. Clearly, the last processor has the most work to do and finishes its execution last.

Now we examine the tasks assigned to μ_{p-1} in detail. Since $|V_i| \approx |V_{i-1}|$, the arithmetic work for $M(V_i, W_i)$ is about the same as the arithmetic work for $M(V_{i-1}, W_{i-1})$. Processors μ_2 and μ_3 start execution at the same time and they finish $M(V_2, W_2)$ and $M(V_3, W_3)$ at about the same time. Then μ_3 waits for the transmission of the transformed block V_2 from μ_2 to μ_3. Therefore, there is no gap or little gap between $M(V_3, W_3)$ and $C(V_2, \mu_2, \mu_3)$. Obviously, there is no gap between $C(V_2, \mu_2, \mu_3)$ and $M(V_2, W_3)$.

$$\begin{tikzcd}
\mu_0 & \mu_1 & \mu_2 & \mu_3 \\
M(V_0, W_0) & M(V_1, W_1) \arrow[d] & M(V_2, W_2) \arrow[d] & M(V_3, W_3) \arrow[d] \\
 & C(V_0, \mu_0, \mu_1) \arrow[d] & C(V_1, \mu_1, \mu_2) \arrow[d] & C(V_2, \mu_2, \mu_3) \arrow[d] \\
 & M(V_0, W_1) & M(V_1, W_2) \arrow[d] & M(V_2, W_3) \arrow[d] \\
 & & C(V_0, \mu_1, \mu_2) \arrow[d] & C(V_1, \mu_2, \mu_3) \arrow[d] \\
 & & M(V_0, W_2) & M(V_1, W_3) \arrow[d] \\
 & & & C(V_0, \mu_2, \mu_3) \arrow[d] \\
 & & & M(V_0, W_3)
\end{tikzcd}$$

FIG. 9. *Tasks for merging V into W on four processors.*

It can be easily shown that the arithmetic work in $M(V_i, W_j)$ is slightly greater than the arithmetic work in $M(V_{i-1}, W_{j-1})$. Once $M(V_2, W_3)$ is finished, the desired message is ready to be received. Hence, there is no gap between $M(V_2, W_3)$ and $C(V_1, \mu_2, \mu_3)$. Similar arguments can be extended to the rest of the tasks executed by μ_3. Therefore, there is no gap or little gap among the tasks executed by μ_3. The discussion for this example is equally valid for a general p.

Since the total number of messages communicated by μ_{p-1} is only $p-1$, the start-up costs for the $p-1$ messages are ignored and the cost for the task $C(V_i, \mu_{p-2}, \mu_{p-1})$ is estimated by the communication volume, i.e., the number of floating-point numbers communicated by the task $C(V_i, \mu_{p-2}, \mu_{p-1})$.

The above analysis shows that the arithmetic and communication complexities for merging V into W on p processors can be obtained by considering the communication volume and the arithmetic work on the last processor μ_{p-1}. Let ℓ be the number of rows in V_{p-1} or W_{p-1}. Then $\ell \approx h/\sqrt{p}$.

Since the total number of rows received by the last processor is $h - \ell$ and each row is of length at most ℓ, the total number of numerical values received by the last processor is bounded above by $(h - \ell)\ell$. Therefore, an upper bound on the communication cost for merging two $h \times h$ upper triangular matrices on p processors is

$$\text{comm}(h, p) = \frac{h^2\tau}{\sqrt{p}} - \frac{h^2\tau}{p}.$$

The work performed by the last processor is bounded above by the arithmetic work required for merging an $(h - \ell) \times \ell$ matrix into an $\ell \times \ell$ upper triangular matrix plus the arithmetic work for merging two $\ell \times \ell$ upper triangular matrices. Therefore, an upper bound on the arithmetic work for merging two $h \times h$ upper triangular matrices on p processors is

$$\text{work}(h, p) = 2\frac{h^3}{p} - \frac{4}{3}\frac{h^3}{p\sqrt{p}} + 2\frac{h^2}{\sqrt{p}} + \frac{4}{3}\frac{h}{\sqrt{p}}.$$

4.2. Analysis of the parallel sparse QR factorization algorithm. Now we analyze the performance of our parallel sparse QR factorization algorithm applied to regular grid problems with nested dissection ordering [7]. Consider a $k \times k$ grid with $(k-1)^2$ small squares. Associated with each square is a set of four equations involving the four variables at the corners of the square. The assembly of these equations results in a sparse overdetermined system of equations $Ax = b$, where A is a $4(k-1)^2 \times k^2$ sparse matrix.

The supernodal elimination tree of a grid problem is a complete binary tree and is mapped onto node processors by the subtree-to-subcube mapping scheme [13]. The frontal matrices are partitioned by our special equal-volume partitioning scheme. A frontal matrix F_K is computed by merging the two update matrices U_1 and U_2 of the children of K. The size of the update matrix of a supernode is at most the size of the frontal matrix of its parent. Assume that the frontal matrix F_K is of order h and it is mapped to p processors. One update matrix U_1 is assembled into F_K and the other update matrix U_2 is merged into F_K. Before U_2 can be merged into F_K, it needs to be appropriately distributed among the p processors. This is equivalent to a parallel assembly of U_2 into F_K. After U_2 is properly distributed among the p processors, U_2 is merged into F_K as in the dense case.

The arithmetic work for merging U_2 into F_K is bounded above by the arithmetic work for merging two $h \times h$ upper triangular matrices in parallel since the size of U_2 is at most the size of F_K. Therefore, an upper bound on the arithmetic work for computing a frontal matrix F_K of order h on p processors is given by work(h, p).

The worst scenario in an assembly operation is that a processor receives $h(h+1)/2p$ numerical values. The communication cost for computing a frontal matrix F_K of order h on p processors is bounded above by the communication cost for assembling U_1 and U_2 into F_K on p processors and the communication cost for merging two $h \times h$ upper triangular matrices on p processors. Therefore, an upper bound on the communication cost for computing a frontal matrix F_K of order h on p processors is

$$\frac{2h(h+1)\tau}{2p} + \text{comm}(h, p) = \frac{h^2\tau}{\sqrt{p}} + \frac{h\tau}{p}.$$

Consider the supernodal elimination tree corresponding to a $k \times k$ grid with nested dissection ordering, where $k = 2^t - 1$ and t is a positive integer. Let $m_j = 2^{t-j} - 1$. The characterization of a heaviest path in the supernodal elimination tree is given in Table 2, where n_i is the size of the frontal matrix on the heaviest path at level i. The root is considered to be at level 1. The reader is referred to [28] for details on the characterization of the supernodal elimination tree associated with a regular grid.

Assume that p is a power of 4. The upper bounds for the arithmetic and communication costs of our algorithm with the special block partition and the conventional row-oriented algorithm are presented in Table 3, where $c_1 = (343 + 250\sqrt{2})/3\sqrt{2}$, $c_2 = (8801\sqrt{2} - 6202)/21\sqrt{2}$, $c_3 = (103 + 72\sqrt{2})/4\sqrt{2}$, and $c_4 = (49 + 50\sqrt{2})/\sqrt{2}$. The results of the row-oriented algorithm are taken from [5]. Under a reasonable assumption that p is less than the number of columns k^2, the communication complexity of our algorithm with the special block partition is much better. The overall time complexity of our algorithm with the special block partition is $1229k^3/4p + O(k^3 \log p/p^{3/2})$ and that of the row-oriented algorithm is $146k^3/3p + 146k^3\tau/12p + O(k^3\tau/p^{3/2})$. If $\tau \geq 22$, the overall complexity of our algorithm with the special block partition is better. Dunigan [6] has shown that the values of τ for

TABLE 2
Characterization of a heaviest path.

Level i	n_i
1	m_0
2	$m_1 + m_0$
3	$m_1 + m_0$
4	$6m_2 + 4$
$2j + 1 (j \geq 2)$	$5m_j + 4$
$2j (j \geq 3)$	$7m_j + 6$

TABLE 3
Arithmetic and communication complexities.

Algorithm	Arithmetic	Communication
special block partition	$\frac{1229}{4}\frac{k^3}{p} - c_1\frac{k^3 \log p}{p^{3/2}} - c_2\frac{k^3}{p^{3/2}} + O(\frac{k^2}{\sqrt{p}})$	$c_3\frac{k^2\tau}{\sqrt{p}} - c_4\frac{k^2\tau}{p} + O(\frac{k\tau \log p}{\sqrt{p}})$
row-oriented partition	$\frac{146}{3}\frac{k^3}{p} + 31\frac{k^2 \log k}{p} + O(\frac{k^2}{p})$	$\frac{146}{12}\frac{k^3\tau}{p} - \frac{371}{12}\frac{k^3\tau}{p^{3/2}} + O(k^2\tau \log p)$

iPSC/2 and iPSC/860 are roughly 59 and 1000, respectively. Note that the communication cost dominates the arithmetic cost in the row-oriented approach.

The results shown in Table 3 are the complexity results of the algorithms which are based on two extreme partitioning schemes in which an $h \times h$ frontal matrix distributed among p processors is partitioned into h and p blocks, respectively. Therefore, it can be expected that the arithmetic complexity of our algorithm based on an arbitrary equal-row or equal-volume partitioning scheme is between the complexity bounds of the row-oriented algorithm and our algorithm with the special block partition. Similarly, the communication complexity of our algorithm based on an arbitrary equal-row or equal-volume partitioning scheme is between the complexity bounds of our algorithm with the special block partition and the row-oriented algorithm.

5. Experimental results. Our parallel multifrontal sparse QR factorization algorithm has been tested on a 32-node iPSC/860 machine for a set of problems including regular grid problems and practical problems with irregular sparsity structure. The practical problems include large and sparse linear least squares problems arising from particle methods for modelling turbulent combustion [25] and a large-problem NIMBUS from the Bramley test set used in [21]. The sparse linear least squares problems for modelling turbulent combustion correspond to three-dimensional $k \times k \times k$ 27-point grids. There are a number of particles associated with each cubic element. A particle corresponds to an equation involving the eight variables at the corners of the cubic element. Assume that a cubic element contains eight particles. The assembly of the equations corresponding to all particles results in a sparse overdetermined system of equations $Ax = b$, where A is an $8(k-1)^3$ by k^3 sparse matrix. The regular grid problems and the problems for modelling turbulent combustion are ordered by the nested dissection ordering [7]. The general sparse problem is ordered by the minimum degree ordering [12].

The characteristics of the test problems are shown in Table 4, where m is the number of rows, n the number of columns, $|A|$ the number of nonzeros in matrix A, and $|R|$ the number of nonzeros in factor R. "GRIDk" represents a $k \times k$ nine-point grid. "TCk" represents a problem for modelling turbulent combustion and corresponds to a three-dimensional $k \times k \times k$ 27-point grid. The QR factorization method described in [14] is used to solve these sparse linear least squares problems on an Intel iPSC/860 machine. The sparse QR factorization is computed by our parallel multifrontal approach described in §3. Both the equal-row partitioning scheme

TABLE 4
Characteristics of the test problems.

Problem	m	n	$\|A\|$	$\|R\|$
GRID100	39204	10000	156816	297953
GRID200	158404	40000	633616	1481803
GRID255	258064	65025	1032256	2569386
TC20	54872	8000	438976	1298284
TC24	97336	13824	778688	2806944
TC27	140608	19683	1124864	4665657
NIMBUS	23871	1325	181972	136356

and the revised equal-volume partitioning scheme are used. The orthogonal transformations are applied to the right-hand-side vector b and then discarded. The required parallel-sparse back-substitution algorithm is described in [29, 31]. The column-oriented kernel is used in our implementation.

Let h be the order of a frontal matrix that is assigned to p processors. We explore four instances of the equal-row partitioning scheme with block sizes equal to 1, h/p^2, $s(h, p)$ as defined in §3.2, and h/p, respectively. For convenience, the versions of our parallel sparse QR factorization algorithm, which use these four instances of the equal-row partitioning scheme in their kernels, are referred to as algorithms ERP1, ERP2, ERP, and ERPP, respectively. We explore two instances of the revised equal-volume partitioning scheme with block sizes corresponding to $s(h, p)$ as defined in §3.2 and h/p, respectively. The versions of our parallel sparse QR factorization algorithm, which use these two instances of the revised equal-volume partitioning scheme in their kernels, are referred to as algorithm EVP and EVPP, respectively. Note that ERP1 is the row-oriented approach proposed in [5] and ERP2 is only suggested as a future research topic in [5]. ERP and EVP are algorithms with optimal block partition.

We evaluate the practical performance of these six parallel sparse QR factorization algorithms on a 32-node Intel iPSC/860 machine. All algorithms are implemented in C and double-precision floating-point arithmetic is used. A dense Givens rotation is implemented as a call to the subroutine `drot` in the highly optimized Basic Math Library provided by Intel. No assembler code other than calls to `drot` is used. The compilation optimization level used is −O4.

We have obtained the running times, communication statistics, and load balance results of the six algorithms for all of our test problems. To save space, the communication statistics and load balance results are presented only for regular grid problems GRID200 and GRID255 on 32 processors. The relative performance of the six algorithms for regular grid problems on a smaller number of processors is the same as that on 32 processors. The relative performance of the six algorithms for other problems is similar to that for these two regular grid problems.

In Table 5, messages communicated per processor are tabulated. The column under "Fact Alg" shows the algorithm used in the numerical factorization. The "max," "min," and "avg" under the "Total # Msgs" are the maximum number, minimum number, and average number of messages communicated by a processor including both messages sent and messages received, respectively. The "max" and "min" under the "# Msgs Sent" are the maximum number and minimum number of messages sent by a processor, respectively. The "max" and "min" under the "# Msgs Recv" are the number of messages received by a processor. The average number of messages sent by a processor is the same as the average number of messages received by a processor and is equal to half of the average number of messages communicated by a processor. For both the equal-row partitioning scheme and the revised equal-volume partitioning scheme, the number of messages required decreases as the block size increases. The number of messages required by EVP is less than that required by ERP since EVP

TABLE 5
Message count per processor ($p = 32$).

Test problem	Fact alg	Total # Msgs max	Total # Msgs min	Total # Msgs avg	# Msgs Sent max	# Msgs Sent min	# Msgs Recv max	# Msgs Recv min
GRID200	ERP1	50838	27344	37824.2	25513	13685	25466	13659
	ERP2	5972	5309	5657.6	3002	2648	3005	2650
	ERP	464	327	390.94	239	161	238	159
	ERPP	113	17	67.56	60	5	54	12
	EVP	284	175	226.25	146	86	144	85
	EVPP	94	17	59.62	48	6	46	11
GRID255	ERP1	83440	43165	61365.2	41841	21568	41599	21596
	ERP2	9188	8351	8817.9	4606	4159	4610	4173
	ERP	465	324	387.4	238	161	236	156
	ERPP	111	17	67.3	59	5	55	12
	EVP	331	178	254.4	169	91	170	87
	EVPP	97	17	64.0	54	6	49	11

TABLE 6
Communication volume per processor (scaled by 10^{-3}, $p = 32$).

Test problem	Fact alg	Total Volume max	Total Volume min	Total Volume avg	Volume Sent max	Volume Sent min	Volume Recv max	Volume Recv min	Avg Msg length
GRID200	ERP1	3562.0	2054.0	2781.6	1783.4	1023.5	1778.7	1028.3	0.074
	ERP2	795.6	734.5	762.9	401.0	362.9	394.7	367.9	0.135
	ERP	261.7	197.6	228.7	132.8	96.7	128.8	99.9	0.585
	ERPP	139.9	23.1	94.3	71.7	7.9	68.2	15.2	1.396
	EVP	250.0	180.2	208.6	126.4	86.0	123.7	90.0	0.922
	EVPP	137.3	38.7	100.7	70.2	14.9	67.7	23.8	1.690
GRID255	ERP1	7349.4	4294.3	5751.2	3678.5	2142.0	3670.9	2148.7	0.094
	ERP2	1504.6	1419.4	1460.6	756.0	704.7	750.9	711.5	0.166
	ERP	416.0	311.0	364.3	211.4	152.3	204.6	158.8	0.940
	ERPP	228.6	40.3	154.1	117.2	11.7	111.4	28.6	2.291
	EVP	411.9	302.1	350.9	208.4	147.1	203.5	149.8	1.380
	EVPP	221.0	69.2	166.3	115.5	24.4	108.5	42.8	2.599

partitions a frontal matrix into fewer blocks than ERP does. Similarly, EVPP communicates fewer messages than ERPP.

In Table 6, the volume of messages communicated per processor is shown. The total volume communicated by a processor is defined as the total number of floating-point double-precision numbers communicated by that processor. The volume sent by a processor is defined as the total number of floating-point double-precision numbers sent by the processor. The volume received is similarly defined. The last column in Table 6 is the average length of all messages obtained by dividing the total communication volume by total number of messages. For both the equal-row partitioning scheme and the revised equal-volume partitioning scheme, the required communication volume decreases as the block size increases.

Consider Tables 5 and 6 together. Clearly, ERP1 and ERP2 require excessive messages and excessive communication volume. For an $h \times h$ frontal matrix partitioned among p processors, the block size for ERP2 is $s = h/p^2$. For reasonable large problem, s is too small. For instance, the root frontal matrix of TC27 is 729 × 729 and is partitioned among 32 processors. Therefore, s is less than one and is set to one in our experiment. The order of the six algorithms in decreasing communication requirements is ERP1, ERP2, ERP, EVP, ERPP, and EVPP. EVPP and EVP are better than ERPP and ERP in terms of communication costs, respectively. The four algorithms ERP, ERPP, EVP, and EVPP communicate relatively long messages. This feature is desirable for distributed-memory multiprocessors

TABLE 7
Load distribution among processors ($p = 32$).

Problem	Fact alg	Max/Avg	Min/Avg
GRID200	ERP1	1.219	0.717
	ERP2	1.290	0.726
	ERP	1.260	0.707
	ERPP	1.514	0.428
	EVP	1.245	0.731
	EVPP	1.313	0.540
GRID255	ERP1	1.230	0.720
	ERP2	1.310	0.705
	ERP	1.273	0.690
	ERPP	1.538	0.412
	EVP	1.254	0.700
	EVPP	1.323	0.531

since the overhead associated with short messages is relatively higher than that associated with long messages.

Load balance results are shown in Table 7. "Max/Avg" is the ratio of the maximum number of floating-point operations performed by a processor to the average number of floating-point operations performed by a processor during the numerical factorization. "Min/Avg" is the ratio of the minimum number of floating-point operations performed by a processor to the average number of floating-point operations performed by a processor during the numerical factorization. A floating-point operation or flop is either a multiplicative or an additive operation. The number of flops performed by a processor is obtained by counting the actual number of flops performed by that processor during the numerical factorization. Roughly, the order of the six algorithms from the least balanced workload distribution to the most balanced workload distribution is ERPP, EVPP, ERP2, ERP, EVP, and ERP1. The difference in workload distribution among ERP2, ERP, EVP, and ERP1 is relatively small. However, the difference between these four algorithms and ERPP or EVPP is fairly pronounced. EVPP and EVP produce better load distribution than ERPP and ERP, respectively.

The running times of the three algorithms ERP2, ERP, and EVP for all test problems on a 32-node Intel iPSC/860 are shown in Table 8. To save space, the running times of the three algorithms ERP1, ERPP, and EVPP are shown in Table 9 for the regular grid problems only. The column under "time" is the running time for the numerical factorization in seconds, which is obtained by measuring the time spent on each processor and taking the maximum time spent on a processor. All processors start at the same time. The "mflops" is the execution rate defined as the number of megaflops performed per second during the numerical factorization. The "time" in Tables 8 and 9 includes the time for applying the Givens rotations to the right-hand-side vector b for all six algorithms; i.e., the time for $Q^T b$ is included in the numerical factorization time.

Due to insufficient storage space on node processors, some test problems cannot be run on a small number of processors. If a test problem can be run on $p = 2^i (0 \leq i \leq 5)$ processors, its timing results on p processors are reported. The running time for a problem on one processor is the time spent by the best serial sparse QR factorization algorithm we have for that problem on one processor.

From Tables 8 and 9 we observe that the order of the six algorithms in increasing efficiency is ERP1, ERP2, ERPP, EVPP, ERP, and EVP. These timing results are in agreement with the communication and workload distribution statistics shown in Tables 5, 6, and 7. Because of the excessive communication requirements, ERP1 and ERP2 are significantly worse than the other four algorithms in performance. ERPP and EVPP are inferior to ERP and EVP. Since

TABLE 8
Performance of the numeric phase of the parallel sparse QR factorization algorithms on an iPSC/860.

Test	No.	ERP2		ERP		EVP	
problem	procs	time	mflops	time	mflops	time	mflops
GRID100	1	17.165	3.971	17.165	3.971	17.165	3.971
	2	8.651	7.878	8.750	7.789	8.780	7.763
	4	4.450	15.316	4.460	15.281	4.487	15.189
	8	2.790	24.428	2.479	27.493	2.487	27.404
	16	1.760	38.724	1.589	42.892	1.446	47.133
	32	1.384	49.245	1.115	61.125	0.932	73.128
GRID200	8	13.567	37.780	13.048	39.283	13.074	39.205
	16	8.754	58.552	7.807	65.655	7.389	69.369
	32	6.689	76.628	4.992	102.678	4.270	120.039
GRID255	32	12.042	85.448	8.124	126.657	7.134	144.234
TC20	8	26.646	53.094	25.941	54.537	24.261	58.314
	16	21.686	65.238	15.454	91.546	13.747	102.913
	32	16.663	84.904	10.123	139.756	8.665	163.272
TC24	16	47.042	86.840	40.040	102.026	34.431	118.646
	32	38.662	105.662	25.608	159.525	21.316	191.645
TC27	32	63.256	127.571	43.766	184.38	35.946	224.494
NIMBUS	1	72.336	8.210	72.336	8.210	72.336	8.210
	2	40.319	14.824	40.277	14.840	40.160	14.883
	4	42.263	15.137	42.624	15.008	39.960	16.009
	8	32.383	20.145	32.660	19.974	29.697	21.967
	16	32.430	20.317	25.247	26.098	23.632	27.881
	32	39.677	16.883	23.874	28.058	22.362	29.955

TABLE 9
Performance of the numeric phase of the parallel sparse QR factorization algorithms on an iPSC/860.

Test	No.	ERP1		ERPP		EVPP	
problem	procs	time	mflops	time	mflops	time	mflops
GRID100	1	17.165	3.971	17.165	3.971	17.165	3.971
	2	9.620	7.085	8.810	7.736	8.802	7.743
	4	6.699	10.174	4.599	14.819	4.583	14.871
	8	5.500	12.392	2.692	25.318	2.661	25.612
	16	4.069	16.750	1.642	41.507	1.540	44.256
	32	3.072	22.186	1.056	64.541	1.019	66.884
GRID200	8	31.591	16.225	14.760	34.727	14.036	36.518
	16	24.191	21.188	8.845	57.950	8.149	62.899
	32	17.712	28.939	5.436	94.291	5.072	101.058
GRID255	32	34.295	30.003	9.740	105.643	8.831	116.517

EVP requires fewer messages and less communication volume than ERP, and EVP produces a more balanced workload distribution than ERP, EVP is more efficient than ERP. Similarly, EVPP is more efficient than ERPP. It is worth noting that EVPP looks promising. One of the advantages of EVPP or ERPP is that it is not necessary to determine the optimal block partitions for a new machine. The performance of all six algorithms for the general sparse problem NIMBUS is worse than that for regular grid problems because of the unbalanced supernodal elimination tree associated with NIMBUS. Speedups for problem GRID100 are shown in Table 10.

6. Concluding remarks. We have described an efficient block-oriented approach to parallel multifrontal sparse QR factorization on distributed-memory multiprocessors. We have proposed novel block schemes for partitioning frontal matrices and explored a spectrum of

TABLE 10
Speedups on problem GRID100.

No. procs	ERP1	ERP2	ERP	ERPP	EVP	EVPP
2	1.78	1.98	1.96	1.95	1.96	1.95
4	2.56	3.86	3.85	3.73	3.82	3.75
8	3.12	6.15	6.92	6.38	6.90	6.45
16	4.22	9.75	10.80	10.45	11.87	11.15
32	5.59	12.40	15.39	16.25	18.42	16.84

block partitioning schemes ranging from the conventional single-row partition to the block partition with maximal block size. Our block-oriented approach results in an efficient parallel sparse QR factorization algorithm with low communication costs, relatively balanced arithmetic work distribution among processors, and a fast parallel-numerical kernel due to the enhanced cache utilization. We have presented both complexity results of our parallel algorithm for regular grid problems and experimental results of an implementation of our parallel algorithm on an Intel iPSC/860 machine. We have demonstrated that our new block-oriented algorithm achieves significant improvement in performance over the conventional nonblock algorithm [5].

Our approach is especially suitable for machines that communicate long messages much more efficiently than they communicate short messages, such as the IBM SP1 system, since relatively long messages are used in our algorithm. An implementation of our parallel sparse QR factorization algorithm on an IBM SP1 system is currently being pursued. We are also extending our algorithm to compute the QR factorization of a sparse matrix with rank deficiency in parallel.

In the implementation of our parallel sparse QR factorization algorithm, the orthogonal matrix Q is not computed since it is often too expensive to store Q in the main memory when A is large. A sparse linear least squares problem $\min_x \|Ax - b\|$ is solved by applying the orthogonal transformations to the right-hand-side vector b during the sparse QR factorization. This approach is numerically backward stable. However, the right-hand-side vector must be available before A is factorized. The method of corrected seminormal equations (CSNE) proposed by Björck [4] can be used to handle new right sides without storing Q. Björck [4] has shown that the CSNE method is, in general, as accurate as the QR factorization method. However, it is not always backward stable and may not be accurate for "stiff" problems. A parallel multifrontal algorithm for solving sparse linear least squares problems on distributed-memory multiprocessors based on the CSNE method is described in [31].

Acknowledgments. I would like to thank Thomas F. Coleman for many discussions relating to this work and for his helpful comments and suggestions on the manuscript. I would also like to thank the referees for their constructive comments and suggestions. Numerical experiments were performed on a 32-node Intel iPSC/860 machine in the Computer Science Department, Cornell University.

REFERENCES

[1] C. ASHCRAFT, *A Vector Implementation of the Multifrontal Method for Large Sparse, Symmetric Positive Definite Linear Systems*, Tech. Report ECA-TR-51, Boeing Computer Services, Seattle, WA, 1987.

[2] C. ASHCRAFT AND R. G. GRIMES, *The influence of relaxed supernode partitions on the multifrontal method*, ACM Trans. Math. Software, 15 (1989), pp. 291–309.

[3] C. ASHCRAFT, R. G. GRIMES, J. G. LEWIS, B. W. PEYTON, AND H. D. SIMON, *Progress in sparse matrix methods for large linear systems on vector supercomputers*, Internat. J. Supercomputer Appl., 1 (1987), pp. 10–30.

[4] Å. Björck, *Stability analysis of the method of seminormal equations for linear least squares problems*, Linear Algebra Appl., 88/89 (1987), pp. 31-48.

[5] E. Chu and J. A. George, *Sparse orthogonal decomposition on a hypercube multiprocessor*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 453-465.

[6] T. H. Dunigan, *Performance of the Intel iPSC/860 Hypercube*, Tech. Report 11491, Mathematical Sciences Section, Oak Ridge National Laboratory, Oak Ridge, TN, 1990.

[7] J. A. George, *Nested dissection of a regular finite element mesh*, SIAM J. Numer. Anal., 10 (1973), pp. 345-363.

[8] J. A. George and M. T. Heath, *Solution of sparse linear least squares problems using Givens rotations*, Linear Algebra Appl., 34 (1980), pp. 69-83.

[9] J. A. George and J. W. H. Liu, *An optimal algorithm for symbolic factorization of symmetric matrices*, SIAM J. Comput., 9 (1980), pp. 583-593.

[10] ———, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[11] ———, *Householder reflections versus Givens rotations in sparse orthogonal decomposition*, Linear Algebra Appl., 88/89 (1987), pp. 223-238.

[12] ———, *The evolution of the minimum degree algorithm*, SIAM Rev., 31 (1989), pp. 1-19.

[13] J. A. George, J. W. H. Liu, and E. G. Y. Ng, *Communication results for parallel sparse Cholesky factorization on a hypercube*, Parallel Comput., 10 (1989), pp. 287-298.

[14] G. Golub, *Numerical methods for solving linear least squares problems*, Numer. Math., 7 (1965), pp. 206-216.

[15] Intel, *iPSC/2 and iPSC/860 User's Guide*, Intel SSD, Beaverton, OR, 1991.

[16] S. G. Kratzer, *Massively Parallel Sparse Matrix Computations*, Tech. Report SRC-TR-90-008, Supercomputing Research Center, Bowie, MD, February 1990.

[17] J. G. Lewis, D. J. Pierce, and D. K. Wah, *Multifrontal Householder QR Factorization*, Tech. Report ECA-TR-127, Boeing Computer Services, Seattle, WA, November 1989.

[18] J. W. H. Liu, *A compact row storage scheme for Cholesky factors using elimination trees*, ACM Trans. Math. Software, 12 (1986), pp. 127-148.

[19] ———, *On general row merging schemes for sparse Givens transformations*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 1190-1211.

[20] ———, *The multifrontal method for sparse matrix solution: Theory and practice*, SIAM Rev., 34 (1992), pp. 82-109.

[21] S. Lu and J. L. Barlow, *Parallel computation of orthogonal factors of sparse matrices*, in Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing, Society for Industrial and Applied Mathematics, Philadelphia, 1993, pp. 486-490.

[22] P. Matstoms, *The Multifrontal Solution of Sparse Linear Least Squares Problems*, Ph.D. thesis, Linköping University, Sweden, 1991.

[23] D. J. Pierce and J. G. Lewis, *Sparse Rank Revealing QR Factorization*, Tech. Report MEA-TR-193, Boeing Computer Services, Seattle, WA, May 1992.

[24] P. E. Plassmann, *Sparse Jacobian estimation and factorization on a multiprocessor*, in Large-Scale Numerical Optimization, T. F. Coleman and Y. Li, eds., Society for Industrial and Applied Mathematics, Philadelphia, 1990, pp. 152-179.

[25] S. B. Pope, *Particle Methods for Turbulent Combustion*, private communication, 1991.

[26] A. Pothen and C. Sun, *A mapping algorithm for parallel sparse Cholesky factorization*, SIAM J. Sci. Comput., 14 (1993), pp. 1253-1257.

[27] P. Raghavan, *Distributed Sparse Matrix Factorization: QR and Cholesky Decompositions*, Ph.D. thesis, Pennsylvania State University, University Park, PA, 1991.

[28] C. Sun, *Some Applications of Clique Trees to Sparse Cholesky Factorizations*, Ph.D. thesis, Pennsylvania State University, University Park, PA, 1991.

[29] ———, *Efficient Parallel Solutions of Large Sparse SPD Systems on Distributed-memory Multiprocessors*, Tech. Report CTC92TR102, Advanced Computing Research Institute, Center for Theory and Simulation in Science and Engineering, Cornell University, Ithaca, NY, August 1992.

[30] ———, *Parallel Sparse Orthogonal Factorization on Distributed-memory Multiprocessors*, Tech. Report CTC93TR162, Advanced Computing Research Institute, Center for Theory and Simulation in Science and Engineering, Cornell University, Ithaca, NY, December 1993.

[31] ———, *Parallel Multifrontal Solution of Sparse Linear Least Squares Problems on Distributed-memory Multiprocessors*, Tech. Report CTC94TR185, Advanced Computing Research Institute, Center for Theory and Simulation in Science and Engineering, Cornell University, Ithaca, NY, July 1994.

SIAM J. SCI. COMPUT.
Vol. 17, No. 3, pp. 686-698, May 1996

A PARALLEL ALGORITHM FOR THE SYLVESTER OBSERVER EQUATION*

CHRISTIAN H. BISCHOF[†], BISWA NATH DATTA[‡], AND AVIJIT PURKAYASTHA[§]

Abstract. We present a new algorithm for solving the Sylvester observer equation arising in the context of the Luenberger observer. The algorithm embodies two main computational phases: the solution of several independent equation systems and a series of matrix-matrix multiplications. The algorithm is, thus, well suited for parallel and high-performance computing. By reducing the coefficient matrix A to lower-Hessenberg form, one can implement the algorithm efficiently, with few floating-point operations and little workspace. The algorithm has been successfully implemented on a CRAY C90. A comparison, both theoretical and experimental, has been made with the well-known Hessenberg-Schur algorithm which solves an arbitrary Sylvester equation. Our theoretical analysis and experimental results confirm the superiority of the proposed algorithm, both in efficiency and speed, over the Hessenberg-Schur algorithm.

Key words. Sylvester observer equation, parallel algorithm, orthogonal factorization, shared-memory parallelism, Hessenberg-Schur algorithm

AMS subject classifications. 15A06, 15A21, 15A23, 65F25, 93B

1. Introduction. The Luenberger observer problem (see [12]) for the control system

$$\begin{aligned} \dot{x} &= Ax + Bu, \\ y &= Cx \end{aligned} \tag{1}$$

arises frequently in control theory. Its solution leads to a Sylvester-type matrix equation:

$$AX - XH = CV. \tag{2}$$

In contrast with the usual Sylvester equation, here only A and C are given, while X, H, and V are to be chosen to satisfy certain requirements. We call (2) the *Sylvester observer equation.* The requirements for choosing H and V are as follows:

(3)

- H must be stable; that is, all the eigenvalues of H should have negative real parts.
- The spectrum of H must be disjoint from that of A (to ensure that X is the unique solution of (2)).
- V must be such that (H^t, V^t) be controllable; that is, the matrix

$$\begin{bmatrix} V^t, & H^tV^t, \dots, (H^t)^{n-1}V^t \end{bmatrix}$$

has rank n.

Since we are free to choose H as long as it satisfies the above criteria, we can choose it as a block upper-Hessenberg matrix with a suitable preassigned spectrum. It can then be shown quite easily that, for this particular structure of H, and

$$V = \begin{pmatrix} 0, & 0, \dots, & 0, & I \end{pmatrix},$$

*Received by the editors December 5, 1991; accepted for publication (in revised form) January 4, 1994.

[†] Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439 (bischof@msc.anl.gov). The work of this author was supported by the Office of Scientific Computing, U.S. Department of Energy, under contract W-31-109-Eng-38.

[‡] Department of Mathematical Sciences, Northern Illinois University, Dekalb, IL 60115. The work of this author was partially supported by a grant from the NSF under contract DMS-9212629.

[§] Department of Mathematics, University of Puerto Rico, Mayagüez, Mayagüez, PR 00681-5000. The work of this author was partially supported by a grant from the NSF under contract RII-8905080.

partitioned conformally, (H^t, V^t) is controllable. This choice of V then reduces (2) to the form

$$AX - XH = \begin{pmatrix} 0, & 0, & \ldots, & 0, & C \end{pmatrix}, \tag{4}$$

where A is a given an $n \times n$ matrix, C is an $n \times r$ matrix, H is a chosen $n \times n$ block upper-Hessenberg matrix with a preassigned spectrum, and X is an $n \times n$ matrix to be constructed. Because we can always choose V in this fashion, we simplified the notation in (2) by using C as shorthand for $C(:, n - r + 1 : n)$. It should be clear from the context whether the whole matrix or only its last r columns are meant.

If W is an invertible matrix, (2) is equivalent to

$$(WAW^{-1})(WX) - (WX)H = (WC)V, \tag{5}$$

and so one can reduce the complexity of the problem by applying a suitably chosen similarity transformation to A. For example, if we were to apply the well-known Hessenberg-Schur method developed by Golub, Nash, and Van Loan [10] for the usual Sylvester equation, to the Sylvester observer equation (4), A would be reduced to Hessenberg form and H to real Schur form (RSF). The RSF of a matrix is a quasi-triangular matrix in which the diagonal entries are either 1×1 or 2×2 matrices (see [11]). Because the matrix H can be chosen for the Sylvester observer equation, one can choose it in RSF with a desired set of eigenvalues on the diagonal and then easily solve for the columns of X. This approach, though numerically effective, does not offer as good a potential for parallelism as the proposed method. A more detailed discussion of this method, from the point of view of solving the Sylvester observer equation, is given in §5.

Another possible approach is the method suggested by Van Dooren [9]. It uses an observer-Hessenberg form for the pair (A, C) in which both A and C are put in certain condensed forms. This approach also requires knowledge of the eigenvalues of the matrix A and is recursive. Like the Hessenberg-Schur method, it computes the columns of X sequentially and does not offer much scope for the exploitation of parallelism.

Yet another approach, based on Arnoldi's method, has recently been proposed by Datta and Saad [3] for the case where C is a vector. It constructs an orthonormal solution to equation (4). The method is suitable for large and sparse problems but does not offer much scope for parallelism.

In this paper, we present a simple yet efficient method for solving (4) which is well suited for parallel and high-performance computers. The method is a block generalization of Datta's method [2] for the case when $r = 1$. In the case where C is $n \times r$, $r > 1$, our method entails solving a total of n independent systems of equations to compute the first r columns of X, and then obtaining the other columns of X, r at a time, essentially through matrix-matrix multiplications. Like the Hessenberg-Schur method, our approach assumes that A is a Hessenberg matrix, and we will not concern ourselves with the reduction of A to Hessenberg form, or the backtransformation of the solution. That is, unless otherwise noted, we assume in the sequel that A in (4) is a lower-Hessenberg matrix. We also note that in our approach, H will be chosen to be a block lower-bidiagonal matrix, with the blocks being diagonal themselves.

We also point out that parallel algorithms for control problems are virtually nonexistent, with only a few algorithms being proposed in recent years (for references to these algorithms, see the recent survey papers of Datta [4] and [5]). The need for expanded research in this area has been clearly outlined in a recent panel report [13].

The outline of the paper is as follows. In the next section we present the algorithm and prove its correctness. In §3 we describe how this algorithm can be implemented efficiently. We show that by initially reducing A to lower-Hessenberg form, and by employing an orthogonal reduction to solve the equation systems, we can fully exploit parallelism in the solution of the independent equation systems, while requiring little additional workspace. In §4 we report on results obtained on a CRAY C90 shared-memory multiprocessor. In §5 we show how to modify the Hessenberg-Schur method for solving the Sylvester observer equation. Results of its parallel implementation are also presented as a means of comparison with the proposed method. Last, we summarize our results and outline directions of further research.

2. The algorithm. In this section we present our algorithm for solving the Sylvester observer equation and prove its correctness. To repeat, we are trying to solve

$$AX - XH = (0, C), \tag{6}$$

where A and C are given $n \times n$ and $n \times r$ matrices, respectively; X is the sought-after $n \times n$ solution matrix; and H is an $n \times n$ matrix that we can choose as long as it satisfies the requirements of (3). In our algorithm we choose

$$H = \begin{bmatrix} \Lambda_{11} & & & & \\ \Lambda_{21} & \ddots & & & \\ & \ddots & \ddots & & \\ & & \ddots & \ddots & \\ & & & \Lambda_{k,k-1} & \Lambda_{kk} \end{bmatrix}. \tag{7}$$

Let

$$(\lambda_{11}, \lambda_{12}, \ldots, \lambda_{1r}), \ldots, (\lambda_{k1}, \lambda_{k2}, \ldots, \lambda_{kr})$$

be the eigenvalues assigned to the diagonal blocks $\Lambda_{11}, \ldots, \Lambda_{kk}$ of the block bidiagonal matrix H with off-diagonal blocks $\Lambda_{i,i-1}$. All of the blocks are of size $r \times r$, where r is the number of outputs or the number of columns of C, and k is the number of blocks such that $rk = n$. The subdiagonals $\Lambda_{i,i-1}$ containing the scaling factors of the ith block X_i will be computed as a byproduct of our solution algorithm such that (6) holds. We partition X conformally such that $X = (X_1, \ldots, X_k)$, where $X_i = (x_1^{(i)}, \ldots, x_r^{(i)})$, and let $C = (c_1, \ldots, c_r)$. In Figure 1 we give our algorithm to solve (6). We now prove the correctness of the algorithm.

THEOREM 2.1. *The algorithm in Figure* 1 *computes the solution of the Sylvester observer equation* (6).

Proof. First, we notice from the block-Hessenberg structure of H in (6) that the blocks X_i of the actual solution X are obtained from the blocks $\hat{X}_i$ of the computed solution $\hat{X}$ by the relation

$$X_{i+1}\Lambda_{i+1,i} = \hat{X}_{i+1} = AX_i - X_i\Lambda_{ii}, i = 1, \ldots, k-1,$$

where $\Lambda_{i+1,i}$ of H contain the 2-norm of the columns of the computed solution $\hat{X}_{i+1}$; that is, $\Lambda_{i+1,i} = \text{diag}(\|\hat{x}_1^{(i+1)}\|_2, \ldots, \|\hat{x}_r^{(i+1)}\|_2)$. Then the first two-block columns of (6) are related by the relation

$$\hat{X}_2 = X_2\Lambda_{2,1} = AX_1 - X_1\Lambda_{1,1}. \tag{8}$$

So if $x_j^{(2)}$ denotes the jth column of X_2, then

$$\hat{x}_j^{(2)} = \|\hat{x}_j^{(2)}\|_2 x_j^{(2)} = (A - \lambda_{1,j}I)x_j^{(1)} \tag{9}$$

```
Compute $X_1$, the first block of $X$:
    For $i = 1, \dots, r$ do
        For $j = 1, \dots, k$ do
            Solve $(A - \lambda_{j,i} I) y_j = c_i$ for $y_j$
            $\alpha_j = (\prod_{\substack{l=1 \\ l \neq j}}^{k} (\lambda_{ji} - \lambda_{li}))^{-1}$
        Enddo
        $x_i^{(1)} = \sum_{j=1}^{k} \alpha_j y_j$
    Enddo

Compute $X_2, \dots, X_k$, the remaining blocks of $X$:
    For $i = 1, 2, \dots, k - 1$ do
        $\hat{X}_{i+1} = A X_i - X_i \Lambda_{ii}$
        $\Lambda_{i+1,i} = \text{diag}(\|\hat{x}_1^{(i+1)}\|_2, \dots, \|\hat{x}_r^{(i+1)}\|_2)$
        $X_{i+1} = \hat{X}_{i+1} \Lambda_{i+1,i}^{-1}$
    End Do
```

FIG. 1. *Algorithm for the solution of the Sylvester observer equation.*

or

$$x_j^{(2)} = \frac{1}{\|\hat{x}_j^{(2)}\|_2} (A - \lambda_{1,j} I) x_j^{(1)}. \tag{10}$$

The remaining blocks $\hat{X}_3, \dots, \hat{X}_k$ then satisfy

$$\hat{X}_{i+1} = X_{i+1} \Lambda_{i+1,i} = A X_i - X_i \Lambda_{i,i}, \; i = 2, \dots, k - 1. \tag{11}$$

If we denote $\hat{x}_j^{(i)}$ to be the jth column of the ith block $\hat{X}_i$, (11) together with the diagonality of $\Lambda_{i,i}$ implies

$$\hat{x}_j^{(i)} = \|\hat{x}_j^{(i)}\|_2 x_j^{(i)} = \frac{(A - \lambda_{i-1,j} I)}{\|\hat{x}_j^{(i-1)}\|_2} \frac{(A - \lambda_{i-2,j} I)}{\|\hat{x}_j^{(i-2)}\|_2} \cdots \frac{(A - \lambda_{2,j} I)}{\|\hat{x}_j^{(2)}\|_2} (A - \lambda_{1,j} I) x_j^{(1)}. \tag{12}$$

Thus, $X_2, \dots, X_k$ are completely determined by X_1.

Then, comparing the last block column in (6), we obtain

$$A X_k - X_k \Lambda_{k,k} = C \tag{13}$$

or

$$(A - \lambda_{k,j} I) x_j^{(k)} = c_j, \; j = 1, \dots, r. \tag{14}$$

Substituting (12) into (14), we obtain

$$\frac{(A - \lambda_{k,j} I)}{\|\hat{x}_j^{(k)}\|_2} \frac{(A - \lambda_{k-1,j} I)}{\|\hat{x}_j^{(k-1)}\|_2} \cdots \frac{(A - \lambda_{2,j} I)}{\|\hat{x}_j^{(2)}\|_2} (A - \lambda_{1,j} I) x_j^{(1)} = c_j, \; j = 1, \dots, r, \tag{15}$$

or

$$(A - \lambda_{k,j} I)(A - \lambda_{k-1,j} I) \dots (A - \lambda_{1,j} I) x_j^{(1)} = \gamma_j c_j, \; j = 1, \dots, r, \tag{16}$$

where

$$\gamma_j = \prod_{i=2}^{k} (\|\hat{x}_j^{(i)}\|_2). \tag{17}$$

Thus, if we solve the polynomial system

$$x_j^{(1)} = p_j(A)^{-1}\gamma_j c_j, \text{ where } p_j(x) = (x - \lambda_{k,j})(x - \lambda_{k-1,j}) \dots (x - \lambda_{1,j}), \quad \text{for } j = 1, \dots, r, \tag{18}$$

we solve (6). □

As (18) indicates, the obvious bottleneck in a practical implementation of the algorithm is the solution of the polynomial system $p_j(A)x_j^{(1)} = \hat{c}_j$, where $\hat{c}_j = \gamma_j c_j$:

$$(A - \lambda_{k,j}I)(A - \lambda_{k-1,j}I) \dots (A - \lambda_{1,j}I)x_j^{(1)} = \hat{c}_j, \; j = 1, \dots, r. \tag{19}$$

The obvious way of solving this system is to successively solve the linear systems

$$(A - \lambda_{i,j}I)y_{i-1} = y_i, \quad i = k, k-1, \dots, 1 \tag{20}$$

with $y_k = c_j$ and $y_1 = x_j^{(1)}$ the final solution. For a very small k, this might not pose any problem. In general, however, it is not satisfactory as it is clearly too expensive. Direct methods for factorization are excluded because A is assumed to be very large. A more effective way, proposed in Datta and Saad [3], is as follows. If we define

$$q(t) = \prod_{j=1}^{k} (t - \lambda_{ji}), \tag{21}$$

the desired solution $x_j^{(1)}$ can be written as

$$x_j^{(1)} = \sum_{i=1}^{k} \frac{(A - \lambda_{i,j}I)^{-1}\hat{c}_j}{q'(\lambda_{i,j})}. \tag{22}$$

In other words, all we need is to solve k independent linear systems

$$(A - \lambda_{j,i}I)y_j = \hat{c}_i, \quad j = 1, \dots, k, \tag{23}$$

and then we obtain $x_j^{(1)}$ as the linear combination

$$x_j^{(1)} = \sum_{i=1}^{k} \alpha_i y_i, \tag{24}$$

where

$$\alpha_i = \left(\prod_{\substack{l=1 \\ l \neq i}}^{k} (\lambda_{i,j} - \lambda_{l,j}) \right)^{-1}. \tag{25}$$

In our experience, this approach for solving the matrix polynomial does not result in any stability problems for the solution of the Sylvester observer equation for different values of the output parameter. This experience is in line with the results in [3].

We note here that the algorithm in Figure 1 will break down if any of the eigenvalues of A and H are close: step 1 or (23) will be singular. By using an appropriate test matrix generator, however, we can guarantee that their spectra do not intersect. In general, we have observed experimentally that the spectra of H and A do not intersect for random matrices, although there is a small probability that they might.

3. Efficient implementation on a shared-memory multiprocessor. In this section we develop an efficient parallel algorithm for the Sylvester observer equation on a shared-memory multiprocessor.

The overall performance of the algorithm hinges critically on efficiently exploiting the apparent parallelism in the computation of X_1, where we have to solve n equation systems $(A - \lambda_{ji}I)y_j = c_i$. Omitting indices for simplicity, we can either use Gaussian elimination or an orthogonal decomposition.

Gaussian Elimination: We decompose $(A - \lambda I)P = LR$, where L is lower triangular, R upper triangular, and P a permutation, then solve the triangular systems $Lw = c$ and $Rz = w$, and, last, undo the permutation by computing $y = Pz$.

Orthogonal Decomposition: Here we have two choices:

QR Factorization: Decompose $(A - \lambda I) = QR$, and solve $Ry = Q^T c$.

LQ Factorization: Decompose $(A - \lambda I) = LQ$, solve $Lz = c$, and compute $y = Q^T z$.

Again, L and R are lower and upper triangular, respectively, and Q is orthogonal. Note that, in contrast to the Gaussian elimination approach, no pivoting is required.

Because we are not interested in the factorization of $(A - \lambda_{ji}I)$ per se, but only in the solution of the equation systems $(A - \lambda_{ji}I)y_j = c_i$, we can perform a forward solve involving a lower triangular matrix L at the same time that we compute L during the factorization. By the same token, we can apply the orthogonal matrix Q to c on the fly if we use the QR factorization.

The drawback is that in both Gaussian elimination and the QR factorization we have to store the upper triangular matrix R, because we cannot start a backsolve involving R before its last element has been computed. Because A is assumed to be dense, both Gaussian elimination and the QR factorization produce a dense upper triangular factor R, which requires a storage of $O(n^2/2)$ words per equation system.

On the other hand, if Q does require little storage, an LQ factorization is well suited. This is the case when A is lower Hessenberg, because then Q can be computed by a sequence of $n-1$ Givens rotations, requiring $O(n)$ storage and only $O(4n^2)$ flops overall. Assuming, as we have done so far, that A is of lower-Hessenberg form, the Sylvester observer equation algorithm in Figure 1 then requires the following:

1. the computation of X_1 by solving a series of systems of equations with lower-Hessenberg coefficient matrices, through an LQ factorization; and
2. the computation of $X_2, \ldots, X_k$ through a recurrence relation involving matrix-matrix products of a lower-Hessenberg and a dense matrix.

As will be seen, the choice of lower-Hessenberg form for A allows us to fully exploit parallelism while keeping extra working storage to a minimum.

3.1. Computing the first block of the solution. To compute the first block X_1 of X, we have to solve n independent systems of linear equations with a lower-Hessenberg coefficient matrix. Each of the r columns c_i of C is the right-hand side for k equation systems.

We have observed experimentally that the conditioning of the problem or the accuracy of our results is not altered appreciably by varying the value of k relative to n, and we assume the ratio $k = n/r$ to be a small constant. This is motivated by the need for obtaining an efficient loop parallelism strategy, as discussed below.

The LQ solver for solving $(A - \lambda I)y = c$ is shown in Figure 2. Here we assume that the vector y holds c on entry and that it contains the solution on exit. The vector e_i is the ith canonical unit vector. Let $P = [1 \rightarrow n, 2 \rightarrow 1, \ldots, n \rightarrow n-1]$ be a left cyclical shift. Then $(A - \lambda I)P = [L, t]$, where L is lower triangular and t is a column vector.

```
Workspace: $w_1, w_2, s$ and $c$, all $n$-vectors.
           $w_1$ and $w_2$ hold the current columns of $L$
           $s$ and $c$ store the $(\sin(\phi_i), \cos(\phi_i))$ pairs.
           We also assume that $A \leftarrow AP$, $P$ being the left cyclic shift.
(1) Initialization: $w_1 = A(:, n) - \lambda e_1, w_2 = A(:, 1) - \lambda e_2$
(2) Compute $L$ and forward solve $Ly = c$ on the fly:
    for $i = 1$ to $n - 1$
      Generate $(c(i), s(i))$ such that $\begin{pmatrix} c(i) & s(i) \\ -s(i) & c(i) \end{pmatrix} \begin{pmatrix} w_1(i) \\ w_2(i) \end{pmatrix} = \begin{pmatrix} 0 \\ \bullet \end{pmatrix}$
      $(w_1(i:n), w_2(i:n)) \leftarrow (w_1(i:n), w_2(i:n)) \begin{pmatrix} c(i) & -s(i) \\ s(i) & c(i) \end{pmatrix}$
      $y(i) \leftarrow y(i)/w_2(i)$
      $y(i+1:n) \leftarrow y(i+1:n) - y(i)w_2(i+1:n)$
      if $i < n - 1$ then
        $w_2(i+1:n) \leftarrow A(i+1:n, i+1) - \lambda e_2$
      endif
    endfor
    $y(n) \leftarrow y(n)/w_1(n)$
(3) Apply $Q^T$ to $y$:
    for $i = n - 1$ downto 1
      $\begin{pmatrix} y(i) \\ y(n) \end{pmatrix} \leftarrow \begin{pmatrix} c(i) & s(i) \\ -s(i) & c(i) \end{pmatrix}^T \begin{pmatrix} y(i) \\ y(n) \end{pmatrix}$
    endfor
(4) Undo Permutation $P$:
    $y \leftarrow Py$
```

FIG. 2. *LQ solver for $(A - \lambda I)y = c$. On entry, the vector y contains the right-hand side c; on exit, it contains the solution of the equation.*

The use of an LQ factorization for a lower-Hessenberg coefficient matrix allows us to compute the orthogonal factor Q, reducing $[L, t]$ to a lower triangular form as a sequence of $n - 1$ Givens rotations:

$$Q = G_1 \cdots G_{n-1},$$

where G_i involves columns i and n. By solving the triangular system $Lz = c$ on the fly, we need to store only two columns of L at any given time. Thus, if we solve p such systems in parallel, we require storage for

$$\begin{array}{ccc} n(n+1)/2 & + & 4pn \\ \text{(for the lower Hessenberg } A) & & \text{(workspace for equation solvers)} \end{array}$$

words. The other alternatives considered before (Gaussian elimination, QR factorization) require $O(pn^2)$ storage instead. Our LQ solver requires roughly $4n^2$ flops to solve an $n \times n$ equation system: $3n^2$ flops for the computation of L and Q, and n^2 flops for the forward solve $Lz = c$. The original solution y is then obtained from $y \leftarrow PQ^Tz$. The algorithm is shown in Figure 2.

The partial drawback of this algorithm is that it employs vector-vector operations, which in general do not perform well on high-performance processors because they require many memory accesses per floating-point operation. (For a discussion of this issue, see, for example, [1,7,8].) If we allow more workspace per processor, we can partially overcome this

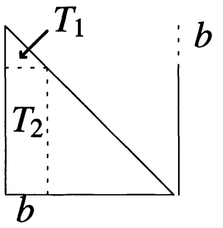


FIG. 3. *Partially completed LQ factorization.*

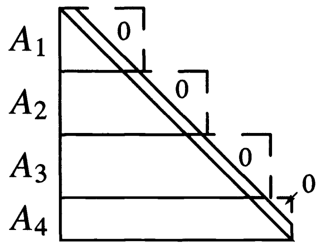


FIG. 4. *Lower-Hessenberg matrix partitioned into four block rows.*

drawback and arrive at a variant that computes the forward solve $Lz = c$ with matrix-vector operations. In particular, if we allow for b columns of workspace for L, we do not have to update the right-hand side y until b columns of L have been updated, while eliminating the first b entries of the last column of A, as shown in Figure 3. After obtaining a strictly lower triangular block, as discussed above, we can now compute the first b entries of y with a triangular solve (BLAS2 routine STRSV),

$$y(1:b) \leftarrow T_1^{-1}\, y(1:b),$$

and update the remaining entries of y with a matrix-vector multiplication (BLAS2 routine SGEMV),

$$y(b+1:n) \leftarrow y(b+1:n) - T_2\, y(1:b).$$

Therefore, instead of computing y one column at a time, the computation is carried out b columns at a time. Thus, in the overall algorithm, roughly 25% of the work is now done by using matrix-vector instead of vector-vector kernels.

3.2. Computing the remaining blocks of *X*. The computational performance of the third step depends on the performance of the matrix-matrix product AX_i. To achieve optimal performance, we should compute this matrix-matrix product in parallel, employing matrix-matrix multiplication as much as possible, while exploiting the lower-Hessenberg structure of A. To this end, we partition A in block rows A_j of width b (not necessarily the same b that is used for the equation solve) and compute AX_i in a block rowwise fashion. That is, we independently compute the first b rows of AX_i by forming A_1X_i, the second b rows by forming A_2X_i, and so on. In general, A_j will be a $b \times (j * b + 1)$ matrix, with a trailing zero upper triangle. A sample partition of A into four block rows is shown in Figure 4. Since in general $b \ll n$, the computations involving zeros will account for only a small portion of the overall computations performed. In particular, for $b = 1$ we employ a matrix-vector multiplication kernel, and no operations with zeros are performed.

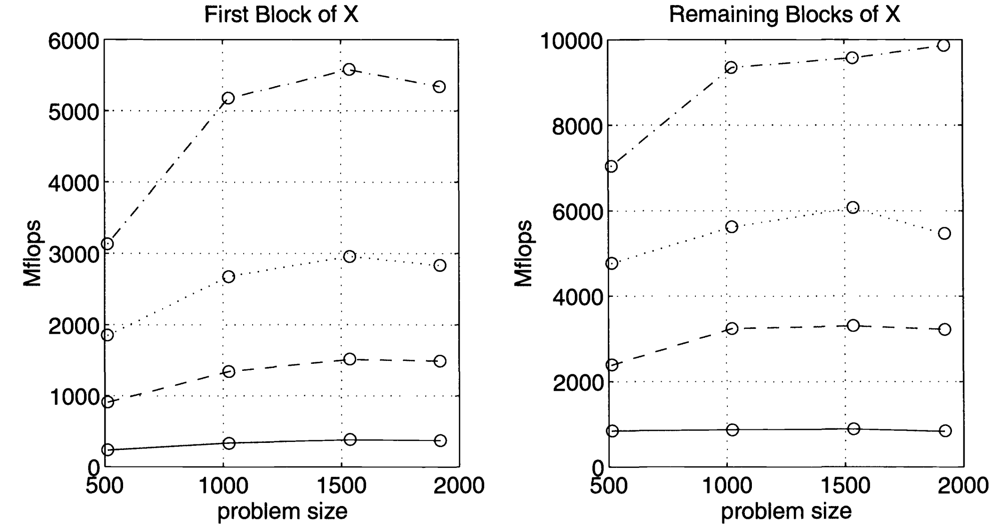


FIG. 5. *Performance of the main kernels of the parallel Sylvester observer equation algorithm on a CRAY C90 (– 1 proc., - - 4 procs., ... 8 procs., -. 16 procs.).*

4. Numerical results. We tested our parallel Sylvester observer equation algorithm on a CRAY C90/16256, a sixteen-processor vector machine with 256 Mwords of shared memory. The CRAY C90 has two sets of vector units per processor, each producing two results per clock-cycle, resulting in a peak performance of 16 Gflops.

As our test problem we generated matrices of dimensions

$$n = 512, 1024, 1536, 1920.$$

To generate lower-Hessenberg matrices with the desired spectrum, we used the LAPACK test matrix generator SLATME [6] to generate a dense nonsymmetric matrix with the desired spectrum, used the LAPACK routine SGEHRD to reduce this matrix to upper Hessenberg form, and then transposed the resulting matrix. In all cases k the number of blocks is 4, so $r = n/4$. We checked the accuracy of our results by computing $fl(C) \equiv A.fl(X_k) - fl(X_k).\Lambda_{k,k}$, which, according to (13), should equal C. In all cases, $fl(C)$ and C agreed to 12 digits. This test is not only cheaper than the usual residual check, but the last block contains the accumulation of all the recurrences and thus is a good indicator of the accuracy of the algorithm.

The BLAS on the CRAY C90 were assembler implementations provided by Cray Research, which exploit multiple CPUs in a fashion that is transparent to the user (unless they are called within a parallel loop, as is the case when we compute the first block of X). Our code obtained the performance and parallel efficiency shown in Figures 5 and 6, respectively.

The plots labeled "First Block of X" and "Remaining Blocks of X" correspond to the two main steps of the Sylvester observer equation algorithm. Figure 7 shows execution rate and efficiency of the two steps combined. In these figures, the solid, dashed, dotted, and dash-dotted lines correspond to runs with 1, 4, 8, and 16 processors, respectively. Efficiency is defined as $\frac{T_1}{T_p \times p} \times 100$, where T_p is the wall-clock time for executing any algorithm on p processors. For all of the segments (i.e., system solutions and matrix products) we get the best results, in general, for blocksize $b = 1$.

Figure 5 shows that the AX_i recursion of the parallel Sylvester observer equation algorithm performs very well. This result is not surprising, because it relies on the highly optimized assembler implementations of the BLAS.

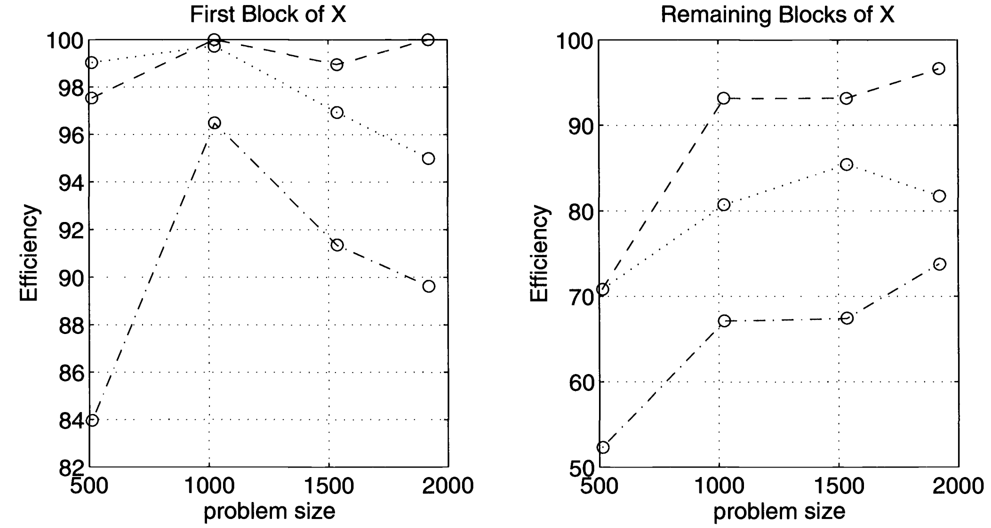


FIG. 6. *Efficiency of the main kernels of the parallel Sylvester observer equation algorithm on a CRAY C90 (- - 4 procs., ... 8 procs., -. 16 procs.).*

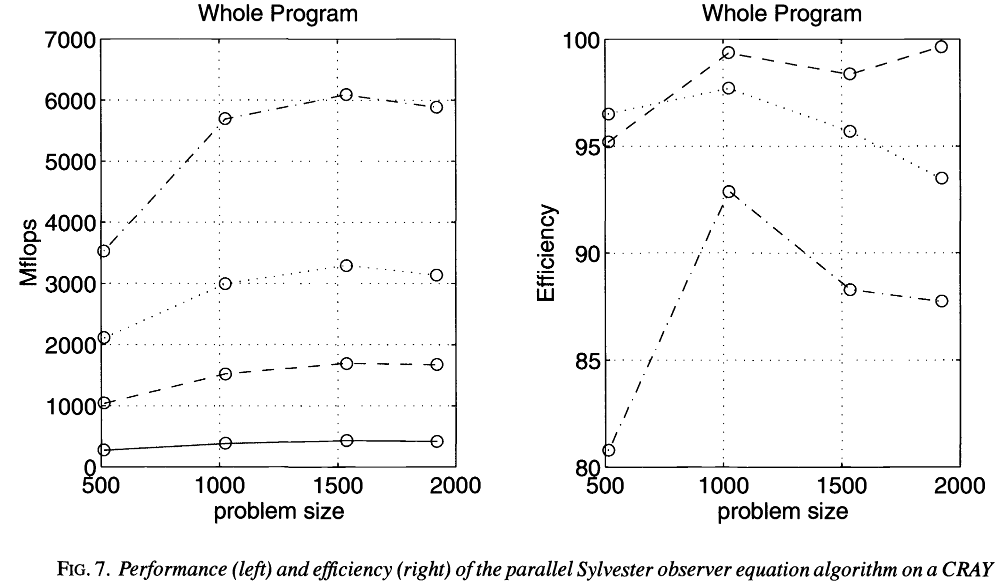


FIG. 7. *Performance (left) and efficiency (right) of the parallel Sylvester observer equation algorithm on a CRAY C90 (– 1 proc., - - 4 procs., ... 8 procs., -. 16 procs.).*

On the other hand, the computation of the first block of X runs much more slowly, at slightly more than half the speed of the other transformations. This is because our on-the-fly LQ solver relies only on BLAS 1 operations for all of its work and the number of systems being solved. The fact that we blocked the computation of y did not result in any improvement on this machine — at most, 8% principally on smaller problems. As a matter of fact, the performance advantage of the blocked version diminished for larger problems because of an increased number of copies in and out of buffers. We also noted that, due to the high internal bandwidth of the C90, the unblocked matrix-matrix multiply does much better than the blocked version, usually performing around 25% faster. However, as the computation

```
Compute $X_k$, the last block of $X$:
    For $i = 1, \ldots, r$ do
        Solve $(A - \lambda_{k,i} I)\hat{x}_i^{(k)} = c_i$
    Enddo
    $\Lambda_{k,k-1} = \mathrm{diag}(\|\hat{x}_1^{(k)}\|_2, \ldots, \|\hat{x}_r^{(k)}\|_2)$
    $X_k = \hat{X}_k \Lambda_{k,k-1}^{-1}$

Solve for $X_{k-1}, \ldots, X_1$, the remaining blocks of $X$:
    For $j = k - 1, \ldots, 1$ do
        For $i = 1, \ldots, r$ do
            Solve $(A - \lambda_{j,i} I)\hat{x}_i^{(j)} = x_i^{(j+1)}$
        Enddo
        $\Lambda_{j,j-1} = \mathrm{diag}(\|\hat{x}_1^{(j)}\|_2, \ldots, \|\hat{x}_r^{(j)}\|_2)$
        $X_j = \hat{X}_j \Lambda_{j,j-1}^{-1}$
    Enddo
```

FIG. 8. *The Hessenberg–Schur method for solution of the Sylvester observer equation.*

of X_1 and the computations of $X_2, \ldots, X_k$ account for roughly 80% and 20%, respectively, of the floating-point operations to be performed, the performance of the overall program very much reflects the performance of the LQ solver. There is little we can do about this situation, since any other solver would require $O(pn^2)$ workspace, which is clearly undesirable.

On the other hand, by exploiting the parallelism inherent in the computation of X, our algorithm scales very well with the number of processors, as the plots in Figures 6 and 7 demonstrate. Not surprisingly, the parallel solution of the equation systems scales the best: because there are many parallel jobs, all with the same computational requirements, the parallel loop is almost perfectly load-balanced. This step is responsible for about half of all floating-point operations, resulting in the high overall parallel efficiency of our code.

5. Comparison of the proposed algorithm with the Hessenberg–Schur method. In this section we make a performance comparison of our proposed parallel algorithm with that of the Hessenberg-Schur method [10] adapted for the solution of the Sylvester observer equation $AX - XH = (0, C)$. The Hessenberg-Schur method solves the usual Sylvester equation $AX - XH = C$, where the matrices A, H, and C are given and X needs to be found. We first show how the Hessenberg-Schur method can be adapted for the solution of the Sylvester observer equation (2).

In the Sylvester observer equation, the matrix H can be chosen arbitrarily as long as it has a preassigned spectrum and its spectrum is different from that of A. Again, we choose H block bidiagonal with $r \times r$ blocks as in (7). Then the Hessenberg-Schur method adapted to the Sylvester observer equation can be described as given by the algorithm in Figure 8.

As assumed in our algorithm, A is also first reduced to Hessenberg form. However, there are key differences in the two approaches. In the proposed new algorithm the first block X_1 is computed first, by solving n systems of linear equations in parallel, and the remaining blocks are then computed recursively using higher-level BLAS operations. In the Hessenberg-Schur algorithm, the last block X_k is computed first, by solving a system of r equations, while the remaining blocks, obtained from the rear, are also obtained by solving systems of r equations at a time. Hence step 2 of Figures 1 and 8 bear little resemblence. In the Hessenberg-Schur approach, it makes sense therefore to parallelize the inner loop, because, as before, r is much larger than k. So we spawn r parallel jobs k times, incurring a greater overhead than in the proposed algorithm, where we spawn n parallel jobs once. Also, there are almost no

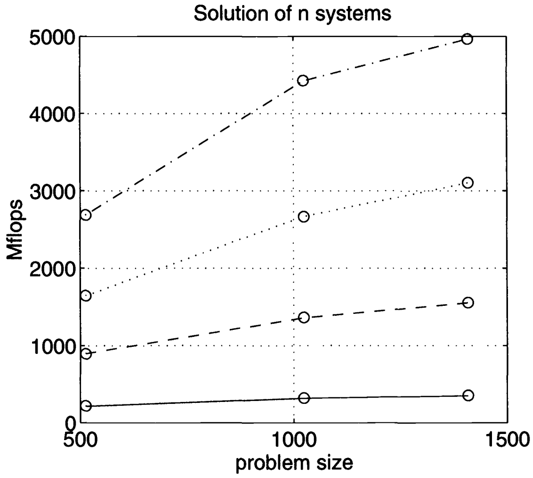


FIG. 9. *Performance of Hessenberg-Schur algorithm on a CRAY C90 (– 1 proc., - - 4 procs., ... 8 procs., -. 16 procs.).*

opportunities for using higher-level BLAS operations in the Hessenberg-Schur algorithm, except in the first step, common to both algorithms, for reducing the matrix to Hessenberg form.

The experimental results on the C90 for the Hessenberg-Schur algorithm is shown in Figure 9. A comparison of the performance graph of Figure 7 with Figure 9 shows that the Hessenberg-Schur algorithm does not perform as well as the new proposed algorithm. For example, for $n = 1536$ and $p = 16$, we obtain around six GFlops with the new algorithm and around five Gflops with the Hessenberg-Schur approach—an improvement of around 20%. Due to memory limitations (the Hessenberg-Schur method seems to require more space to execute), we could not run the case $n = 1920$.

6. Conclusions. In this paper we presented a new parallel algorithm for solving the Sylvester observer equation. The algorithm is simple and relies on standard linear algebra building blocks. The main computational steps are a reduction to Hessenberg form, the solution of a series of independent equation systems, and a recurrence relation based on matrix-matrix multiplies. These attributes, together with the parallelism in the algorithm, are key requirements for an efficient implementation on a shared-memory multiprocessor. By reducing the coefficient matrix to lower-Hessenberg form, we can implement our algorithm with little additional workspace, thereby ensuring that we can solve big problems and that our algorithm scales well with the number of processors. Experimental results on a CRAY C90 show that the algorithm is indeed well suited for a shared-memory multiprocessor. Also, a comparison is made with the well-known Hessenberg-Schur algorithm.

At the moment we are working on a version of this algorithm that is suitable for a distributed-memory multiprocessor. As in our current implementation, the key issue will be an efficient implementation of the parallel equation solves and the limitation of workspace. Research into sparse implementations is also in progress.

Acknowledgments. We are indebted to Bruce Kelly and Moe Jette of the National Energy Research Supercomputer Center at Lawrence Livermore National Laboratory for their dedicated help with the benchmarking of our code. We also thank Klaus Geers of the University of Karlsruhe and Phuong Vu of Cray Research for performing some preliminary benchmark runs for us. We are also indebted to Phuong Vu for suggesting some programming enhancements.

Last, we thank an anonymous referee for suggesting the left cyclic shift reordering in the LQ solver.

REFERENCES

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. DuCroz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK User's Guide*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.

[2] B. N. Datta, *Parallel and large-scale matrix computations in control: Some ideas*, Linear Algebra Appl., 121 (1989), pp. 243-264.

[3] B. N. Datta and Youcef Saad, *Arnoldi methods for large Sylvester-like observer matrix equations, and an associated algorithm for partial spectrum assignment*, Linear Algebra Appl., 154/156 (1991), pp. 225-244.

[4] B. N. Datta, *Parallel algorithms in control theory*, in Proc. IEEE Conference on Decision and Control, 1991, pp. 1700-1704.

[5] ———, *High performance in linear control*, in Proc. SIAM Conference on Parallel Processing, 1993, pp. 274-281.

[6] J. Demmel and A. McKenney, *LAPACK working note 9: A test matrix generation suite*, Preprint MCS-P69-0389, Mathematics and Computer Science Division, Argonne National Laboratory, August 1989.

[7] J. J. Dongarra, F. G. Gustavson, and A. Karp, *Implementing linear algebra algorithms for dense matrices on a vector pipeline machine*, SIAM Rev., 26 (1984), pp. 91-112.

[8] J. Dongarra and S. Hammarling, *Evolution of numerical software for dense linear algebra*, in Evolution of Numerical Software for Dense Linear Algebra, M. G. Cox and S. Hammarline, eds., Oxford University Press, Oxford, UK, 1989.

[9] P. Van Dooren, *Reduced order observers: A new algorithm and proof*, Systems Control Lett., 4 (1984), pp. 243-251.

[10] G. Golub, S. Nash, and C. Van Loan, *A Hessenberg Schur method for the problem $AX - XB = C$*, IEEE Trans. Automat. Control, 6 (1979), pp. 909-913.

[11] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, MD, 1983.

[12] D. Luenberger, *Observing the state of a linear system*, IEEE Trans. Military Electronics, MIL-8 (1964), pp. 74-80.

[13] Report of the panel *Future Directions in Control Theory: A Mathematical Perspective*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1988.

SIAM J. SCI. COMPUT.
Vol. 17, No. 3, pp. 699–713, May 1996

009

PERFORMANCE OF PANEL AND BLOCK APPROACHES TO SPARSE CHOLESKY FACTORIZATION ON THE iPSC/860 AND PARAGON MULTICOMPUTERS*

EDWARD ROTHBERG†

Abstract. Sparse Cholesky factorization has historically achieved extremely low performance on distributed-memory multiprocessors. We believe that three issues must be addressed to improve this situation: (1) parallel factorization methods must be based on more efficient sequential methods; (2) parallel machines must provide higher interprocessor communication bandwidth; and (3) the sparse matrices used to evaluate parallel sparse factorization performance should be more representative of the sizes of matrices people would factor on large parallel machines. This paper demonstrates that all three of these issues have in fact already been addressed. Specifically, (1) single node performance can be improved by moving from a column-oriented approach, where the computational kernel is level 1 BLAS, to either a panel- or block-oriented approach, where the computational kernel is level 3 BLAS; (2) communication hardware has improved dramatically, with new parallel computers (the Intel Paragon system) providing one to two orders of magnitude higher communication bandwidth than previous parallel computers (the Intel iPSC/860 system); and (3) several larger benchmark matrices are now available, and newer parallel machines offer sufficient memory per node to factor these larger matrices. The result of addressing these three issues is extremely high performance on moderately parallel machines. This paper demonstrates performance levels of 650 double-precision Mflops on 32 nodes of the Intel Paragon system, 1 Gflop on 64 nodes, and 1.7 Gflops on 128 nodes. This paper also does a direct performance comparison between the iPSC/860 and Paragon systems, as well as a comparison between panel- and block-oriented approaches to parallel factorization.

Key words. sparse Cholesky factorization, parallel machines, sparse matrices, scalability

AMS subject classifications. 65F05, 65F50

1. Introduction. Sparse Cholesky factorization is an extremely important computation. It arises in a wide variety of application domains, including finite-element analysis, linear programming, and semiconductor simulation. Sparse Cholesky factorization is extremely time consuming and, consequently, there is substantial interest in obtaining higher factorization performance. One seemingly promising approach is the use of distributed-memory multiprocessors.

Unfortunately, parallel sparse Cholesky factorization results on distributed-memory multiprocessors have been disappointing so far [2, 8, 11, 15, 21]. Such machines have not yet been shown to offer substantial advantages over alternative platforms for the computation. Specifically, their performance has not been significantly higher than that of high-end uniprocessors available at the same time, and their performance has not been at all competitive with the performance of vector supercomputers [4, 14, 20].

We believe there are three primary reasons for the disappointing performance of distributed-memory multiprocessors for this computation. The first is that parallel factorization methods have traditionally been based on inefficient sequential methods. Virtually all of these parallel methods distribute columns of the matrix among the processors, leading to a level 1 BLAS computational kernel. Many researchers [1, 4, 14, 17, 18] have demonstrated that sequential sparse factorization methods that are based on level 3 BLAS kernels [5] provide many times the performance of sequential methods that are based on level 1 kernels. This paper considers two approaches to integrating level 3 BLAS operations into the parallel computation. The first, a panel multifrontal method, distributes sets of adjacent columns among the processors. The second, a block fan-out method, distributes rectangular patches.

*Received by the editors May 4, 1994; accepted for publication (in revised form) December 21, 1994. A preliminary version of this paper appeared in *Proceedings of the Scalable High-Performance Computing Conference*, IEEE Comput. Soc. Press, Piscataway, NJ, 1994, pp. 324–333.

†Intel Supercomputer Systems Division, 14924 N.W. Greenbrier Parkway, Beaverton, OR 97006. Current address: Silicon Graphics, Inc., Mountain View, CA 94043 (rothberg@sgi.com).

We demonstrate the advantages of the higher-level kernels by directly comparing performance against previously reported performance numbers for a parallel column method [21] on an iPSC/860 system. The improvements are significant on small machines but fall off for larger machines.

The second reason for the low performance observed in earlier investigations is the fact that earlier distributed-memory multiprocessors did not have sufficient communication bandwidth to support the enormous interprocessor communication demands of parallel sparse Cholesky factorization. Fortunately, multiprocessor communication hardware has improved dramatically. For example, the Intel Paragon system provides one to two orders of magnitude higher communication bandwidth than the earlier Intel iPSC/860 system. We will show that communication bandwidth is now sufficient to support extremely high parallel performance for this computation.

The third factor that limits parallel performance is the size of problems that have traditionally been used as parallel sparse factorization benchmarks. While these problems often require several hundred million floating-point operations to factor, they are still too small to make good use of even a moderate number of processors. This can best be observed by examining the task dependency graphs that result from these sparse problems [18]. The critical paths in the task graphs for these problems are too long to allow more than a small number of processors to be productively employed in the factorization. The limited size of these benchmark matrices has probably been motivated by the small amount of memory available on each node of a parallel machine. Fortunately, parallel machines are now available with substantially more memory per node. It is our belief that the matrices that have been used as benchmarks are unrealistically small, and that the larger matrices we will consider in the latter part of this paper are more representative of the sparse matrices people will want to factor on large parallel machines.

Overall, this paper demonstrates the following results. First, it demonstrates that both panel- and block-oriented methods produce higher performance than column-oriented methods on the iPSC/860 system. However, the improvement falls off with an increasing number of processors, primarily because communication costs begin to dominate. These alternative methods do not significantly reduce these costs. Second, the paper demonstrates that the Paragon system provides substantial performance advantages over the iPSC/860 system. We observe factors of three or more performance improvement on 32 or 64 processors for the traditional benchmark matrices. Third, the paper demonstrates that the block fan-out method produces significantly higher performance than the panel multifrontal method on larger parallel machines. Finally, this paper demonstrates that for large sparse problems, a distributed-memory multiprocessor is capable of producing extremely high factorization performance. We demonstrate factorization performance on the Paragon system in the range of 650 double-precision Mflops on 32 processors, 1 Gflop on 64 processors, and 1.7 Gflops on 128 processors.

We should note that the factorization methods we consider in this paper, the panel multifrontal method and the block fan-out method, have been described and investigated previously [16, 18]. The main contributions of this paper are its direct comparison of these two methods on widely available parallel machines and its direct comparison of the performance of two generations of parallel machines for this important computation.

The organization of this paper is as follows. Section 2 provides a short background on sparse Cholesky factorization and the parallel methods we use to perform the computation. Section 3 describes our experimental environment, including the sparse matrices we use as benchmarks and the machines on which we factor these matrices. Section 4 looks at parallel factorization performance, comparing the performance of the panel multifrontal and block fan-out methods on the iPSC/860 and Paragon systems. Section 5 then considers the performance improvement that comes from factoring larger problems on the Paragon system. Finally, §6 presents a brief discussion, and §7 presents conclusions.

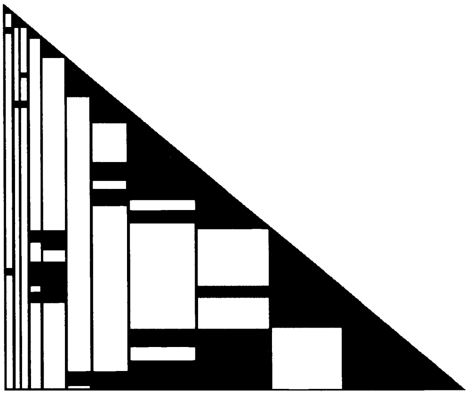


FIG. 1. *Supernodal structure of a sparse matrix.*

2. Sparse Cholesky factorization.

2.1. Computation structure. The goal of sparse Cholesky factorization is to factor a sparse-symmetric positive-definite matrix **A** into the form **A** = $\mathbf{LL}^{\mathrm{T}}$, where **L** is lower triangular with positive diagonal elements. This is typically accomplished in three steps. The first step, *heuristic reordering*, reorders the rows and columns of **A** using a heuristic such as multiple minimum degree [12] or nested dissection [9] to reduce *fill* in the factor matrix. The second step, *symbolic factorization*, performs the factorization symbolically to determine the nonzero structure of the factor matrix after all fill has occurred. This step allocates storage for the factor matrix. The third step, *numerical factorization*, computes the actual numerical values in **L**. We refer the reader to [9] for more information. We concentrate on the numerical factorization step in this paper, since it is by far the most time consuming.

The following pseudocode performs Cholesky factorization:

```
1. Set L = A
2. for k = 1 to n do
3.    for i = k to n do
4.       $L_{ik} := L_{ik}$ / sqrt($L_{kk}$)
5.    for j = k + 1 to n do
6.       for i = j to n do
7.          $L_{ij} := L_{ij} - L_{ik}\ L_{jk}$
```

For a sparse matrix, the factorization would only perform operations on nonzeros in the matrix, and it would only store these nonzeros. Thus, the loops in steps 3, 5, and 6 would iterate over a subset of all possible values (the subset is easily determined from the structure of the sparse matrix).

Sparse factorization is typically expressed in terms of columns of the sparse matrix. Within a column-oriented framework, steps 3 and 4 above can be thought of as a single column division or cdiv(k) operation. This operation divides all nonzeros in a column by the square root of its diagonal. Steps 6 and 7 form a column modification or cmod(j, k) operation. This operation adds a multiple of column k into column j.

2.2. Supernodal structure. A crucial notion for improving the performance of sparse factorization is that of the *supernode* [4]. A supernode is a set of adjacent columns in the sparse factor matrix whose structure consists of a dense triangular block on the diagonal and identical nonzero structures in each column below the diagonal. Supernodes arise in any sparse factorization, and they are typically quite large. Figure 1 gives an example of the structure of a typical factor matrix. The shaded areas represent nonzero values, and the vertical lines delineate the supernodes.

The factorization computation can easily be expressed in terms of supernodes; there are simple supernode analogues for the column-oriented operations described earlier. An sdiv(K) would multiply supernode K by the inverse of its diagonal block. Similarly, an smod(J, K) operation would modify the values in supernode J by those in supernode K. These supernode analogues are rich in level 3 BLAS operations and, as a result, they lead to substantially higher performance than the corresponding column versions.

Performance of the factorization can be further improved by performing *supernode amalgamation* [3] on the factor matrix before the factorization. Supernode amalgamation is a process of identifying zero-valued locations in the matrix that would produce larger supernodes if they were treated as nonzeros and explicitly stored in the sparse representation. All factorization methods we investigate in this paper perform supernode amalgamation before the factorization.

In the context of parallel factorization, supernodes unfortunately represent too coarse of a distribution grain. Some supernodes are so large that mapping them to a single processor would cause load balance problems. The methods investigated in this paper divide supernodes into finer-grain objects. Our panel multifrontal method divides supernodes into panels, which are contiguous sets of columns from within the same supernode. Our block fan-out method divides supernodes into rectangular blocks. We will describe the block decomposition in more detail shortly.

2.3. Parallel sparse Cholesky factorization. We now briefly describe the two parallel factorization methods we consider in this paper, the panel multifrontal method and the block fan-out method. These methods have been described in detail elsewhere [16, 18], so this discussion will omit the details and concentrate only on the aspects of these methods that are relevant for this paper.

2.3.1. Panel multifrontal method. The panel multifrontal method is a simple variant of the parallel column multifrontal method [2, 13]. The multifrontal method is quite complicated, and its details are beyond the scope of this paper. We only discuss its most important components here.

Perhaps the most important computational operation in the parallel *column* multifrontal method is the column modification, or cmod(j, k). Recall that this operation adds a multiple of column k of the matrix into column j. The multifrontal method structures the computation in such a way that, even though columns j and k may have different nonzero structures, the cmod(j, k) operation can be performed using dense matrix operations. For a column method, the appropriate dense matrix operation is a DAXPY.

Another important operation in the parallel column multifrontal method is the column multicast. In a parallel multifrontal method, modifications *from* a particular column k are typically performed by several processors. Thus, once a column k has received all modifications from other columns, it is multicast from its owner (each column has an owner processor) to all processors that perform modifications with k as their source. We use a *proportional mapping* scheme [15] to map columns and column modifications to processors. This scheme does a good job of balancing the computational load among the processors while simultaneously limiting communication volumes.

The panel version of the multifrontal method is quite similar to the column version. The main difference is that the former distributes panels among the processors while the latter distributes columns. The main computational operation becomes a panel modification, pmod(J, K). It is performed as a dense matrix-matrix multiplication (a DGEMM using BLAS

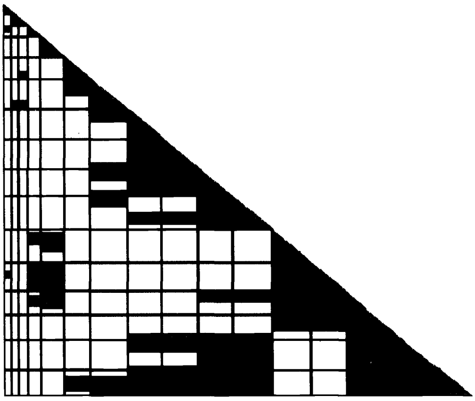


FIG. 2. *Block decomposition of a sparse matrix.*

terminology). The main communication operation is a panel multicast, which multicasts coarser-grain panels instead of columns.

One detail about the panel multifrontal method that we must mention is how supernodes are divided into panels. We do so by choosing a global panel width and dividing all supernodes in the matrix into panels of this width. Since supernode widths are not necessarily multiples of the chosen panel width, there will sometimes be leftover panels. For example, if the global panel width is 16, a supernode containing 60 columns would be divided into three panels of width 16 and one panel of width 12.

2.3.2. Block fan-out method. The details of the block fan-out approach [16] are again beyond the scope of this paper. We just describe the most important issues for this method, which are (1) the decomposition of the sparse matrix into blocks, (2) the operations processors perform on these blocks, and (3) the communication performed by the processors.

The block fan-out method partitions the columns of the matrix $(1, \ldots, n)$ into contiguous subsets $(\{1, \ldots, p_2 - 1\}, \{p_2, \ldots, p_3 - 1\}, \ldots, \{p_N, \ldots, n\})$, where N is the number of subsets. All columns in a subset must be members of the same supernode. An identical partitioning is performed on the rows. A block L_{IJ} (we refer to subsets using capital letters) consists of the set of nonzeros that fall simultaneously in rows $\{p_I, \ldots, p_{I+1} - 1\}$ and columns $\{p_J, \ldots, p_{J+1} - 1\}$. To form these subsets, we again choose a single global block size and partition all supernodes into subsets of this size. Figure 2 gives an example of such a decomposition.

Perhaps the most important operation in the block fan-out method is the block modification. A block modification, or bmod(L_{IJ}, L_{IK}, L_{JK}), modifies one destination block (L_{IJ}) using two source blocks (L_{IK} and L_{JK}). This modification is performed as a dense matrix-matrix multiplication (or DGEMM). Blocks sometimes have sparse structure, which may necessitate sparse operations in the bmod() operation. Such operations are infrequent, however.

To restrict communication volumes, blocks are mapped to processors using a *scatter decomposition* [7]. In this mapping strategy, the processors are thought of as a two-dimensional (2-D) r-by-s-grid. A block L_{IK} is mapped to processor (I mod r, K mod s) in this grid. All block modifications are performed by the processor that owns the destination block. This mapping, combined with the property that a block L_{IK} can only modify blocks in block row I or block column I, leads to the property that L_{IK} need only be sent to the row and column of processors in the processor grid that own row I and column I of blocks. This property limits communication volumes.

We should mention that we do not perform a scatter decomposition on the entire sparse matrix. It is possible to assign a large, contiguous set of columns of the matrix (typically referred to as a *domain* [2]) to each processor. The remainder of the matrix (whose size is still substantial) is then distributed among the processors using a scatter decomposition. The use of domains reduces communication, since the matrix blocks in these domains do not have to be communicated to other processors.

2.3.3. Important issues for parallel methods. We now briefly comment on some of the more important properties of these parallel methods. In particular, we comment on four issues that may affect parallel speedups and relative performance. We will refer back to these issues throughout this paper when we investigate performance of the methods.

The first issue that may affect parallel performance is load imbalance. Recall that the proportional mapping used for the panel multifrontal method does a good job of balancing the computational load. In contrast, the block fan-out method uses a relatively rigid scatter decomposition, which makes load balancing more difficult. The panel multifrontal method will have better overall load balance.

A second issue that may affect performance is communication costs. For a k-by-k 2-D grid problem, which requires $O(k^3)$ floating-point operations to factor, the panel multifrontal method asymptotically generates $O(k^2P)$ communication volume [10, 16]. The block fan-out method generates $O(sqrt(P)k^2 \log P)$ communication. Expressed as computation to communication ratios, these methods perform $O(k/P)$ and $O(k/(sqrt(P) \log P))$ floating-point operations per communicated word, respectively. Similar growth rates apply for other sparse problems. While these growth rates favor the block approach, the measured differences turn out to be small for 64 or fewer processors. For the problems we consider, the difference in communication volumes between the column multifrontal method, the panel multifrontal method, and the block fan-out method is always less than 30%. Furthermore, none of the three methods always produces either the most or the least communication. Communication volume issues therefore do not favor one method over the other in this study. Communication volumes are extremely high, however, and they will affect performance for both methods.

The other important cost associated with communication, the *fixed* cost of sending a message between processors, is not very important for either the panel multifrontal or the block fan-out method. Both methods group nonzeros into sufficiently large-grain messages that the cost of sending a message is dominated by the bandwidth component of the message send cost, not the fixed component.

A third issue that affects performance is the length of the critical path in the factorization computation task graph. This critical path places a lower bound on parallel runtime, or, equivalently, it places an upper bound on parallel performance. For a k-by-k 2-D grid problem using a partition width of B, the critical path for a panel multifrontal method contains $O(k^2B)$ floating-point operations and $O(k^2)$ words of communication. In contrast, the critical path for a block fan-out method contains $O(kB^2)$ operations and $O(kB)$ words of communication. Expressed as maximum parallel speedups, these bounds give $O(k/B)$ best-case speedup for the panel multifrontal method and $O(k^2/B^2)$ speedup for the block fan-out method. In practice, the critical path is much more constraining for the panel multifrontal method.

A fourth issue that affects parallel performance is single node performance. The panel multifrontal and block fan-out methods are both based on level 3 BLAS kernels and they both provide similar single-node performance. For larger numbers of processors, however, the panel and block widths must be decreased to reduce the computational grain and increase available concurrency. This will lead to lower per-node performance than would be obtained for a sequential code, where there is no need to limit the panel or block sizes.

TABLE 1
Benchmark matrices.

Problem name	Equations	NZ in A	NZ in L	Ops for factorization
GRID150	22,500	111,900	774,652	56,428,437
CUBE20	8,000	53,600	1,235,018	302,639,877
BCSSTK15	3,948	117,816	684,655	165,039,042
BCSSTK16	4,884	290,378	773,384	149,105,832
BCSSTK17	10,974	428,650	1,054,402	144,280,005
BCSSTK18	11,948	149,090	738,935	140,919,771
BCSSTK29	13,992	619,488	1,757,352	393,059,150

One thing to note about these four issues is that each becomes less problematic with larger problems. Load balancing becomes easier when there are more panels or blocks to distribute. Computation-to-communication ratios increase with increasing k, reducing relative communication costs. Similarly, maximum parallel speedups due to critical path limits improve with increasing k. Larger problems also allow larger panel or block sizes, which increase per-node performance. We therefore expect higher parallel performance from larger problems.

When we investigate obtained performance in this paper, we will try to explain it in terms of these four factors. Unfortunately, parallel performance cannot be perfectly explained in this way. The intricacies of the parallel machine and of the sparse Cholesky factorization computation make it impossible to account for every millisecond of parallel runtime. We therefore use these factors to provide insight into the observed parallel behavior.

3. Experimental environment.

3.1. Benchmark matrices. Table 1 describes the matrices we use as initial benchmarks. These include some of the larger matrices from the Harwell-Boeing sparse-matrix test set [6]. We also include simple 2-D and 3-D grid problems, which we refer to as GRID and CUBE problems, respectively.

We use double-precision arithmetic for all factorizations in this paper. All matrices are ordered using the multiple minimum degree ordering heuristic [12], with the exception of the GRID and CUBE problems, which are ordered using nested dissection [9]. All Mflop numbers in this paper are computed by dividing the total number of floating-point operations required to factor the problem using a sequential method by the appropriate runtime. Floating-point operation counts do not include operations introduced by the supernode amalgamation.

3.2. Parallel machines. As mentioned earlier, we perform our performance evaluation using the iPSC/860 and Paragon distributed-memory multiprocessors. Both use a message-passing programming model, wherein processors exchange data through explicit send and receive commands.

The iPSC/860 system uses a 40 MHz Intel i860 XR processor at each node. The machine we use has 64 nodes; 8 of these have 64 MBytes of memory and the other 56 have 8 MBytes of memory. The interconnection network has a hypercube topology. The peak bandwidth between two nodes is 2.8 MBytes per second.

The Paragon system uses a 50 MHz i860 XP processor at each node. The machine we use has 140 nodes, all with 32 MBytes of memory. The Paragon system interconnection network is organized as a 2-D mesh. Peak hardware communication bandwidth between two nodes is 200 MBytes per second. The machine we use is running release 1.2 of the operating system, which delivers a peak application message passing bandwidth of 80 MBytes per second. We

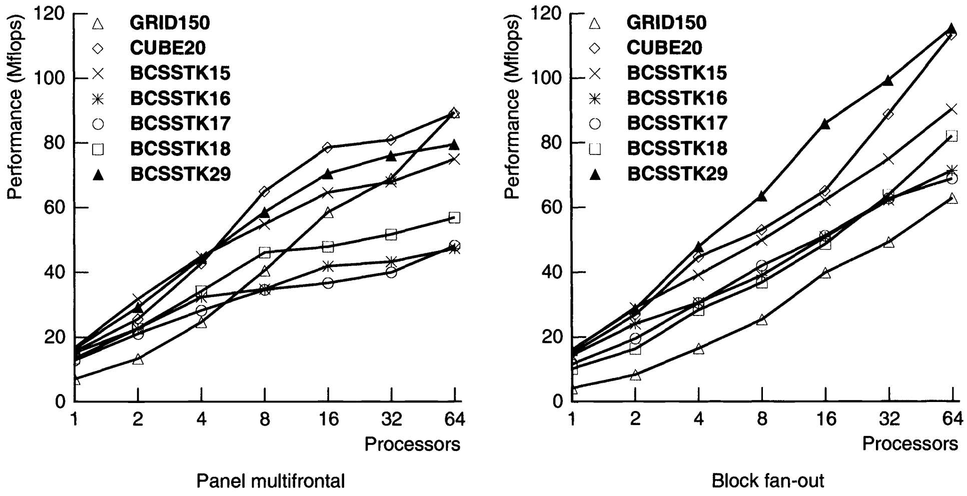


FIG. 3. *Factorization performance on the iPSC/860 system.*

measured roughly 40 MBytes per second of delivered bandwidth for the message sizes that are typical for the factorization. This represents more than a factor of 10 increase in delivered bandwidth over the iPSC/860 system.

It is often convenient to view the costs of communicating data between nodes in terms of the costs of performing floating-point operations on the nodes. If we use computation rates of 20 Mflops for the iPSC/860 system and 25 Mflops for the Paragon system (reasonable approximations, as we will show), then the incremental cost of sending one floating-point datum on the iPSC/860 system is roughly equal to the cost of performing 55 floating- point operations. The corresponding number is 5 floating-point operations on the Paragon system.

3.3. Implementation details. Our implementations of both the panel multifrontal and block fan-out methods are written in the C programming language, although we call the vendor-provided, hand-coded BLAS library routines whenever possible. Note that the BLAS routines on the machines we use cannot work with packed storage formats for triangular or trapezoidal matrices. Consequently, we store an upper-triangular block of zeros at the top of each panel in the panel multifrontal method and at the top of each diagonal block in the block fan-out method. The storage costs associated with these zeros are small.

We use dynamic memory for all important matrix data structures. Panels and blocks are stored as individually allocated objects. Similarly, temporary results (panel modifications in the panel multifrontal method and blocks received from other nodes in the block fan-out method) are stored in dynamically allocated blocks. We measured the costs of managing this dynamic memory and found them to be negligible.

We note that the iPSC/860 and Paragon systems are source-code compatible. We use the identical factorization codes on both. We also note that the Paragon system is a virtual-memory machine. To eliminate paging effects, our codes perform four factorizations in succession and report the best of the four runtimes.

4. Parallel performance.

4.1. iPSC/860. We now look at parallel performance of the panel multifrontal and block fan-out methods on the iPSC/860 system. Figure 3 shows factorization performance on 1 to 64 nodes. For the panel multifrontal method, the numbers in the figure give the best performance obtained using a panel width of 16, 24, or 32. For the block fan-out method, the numbers

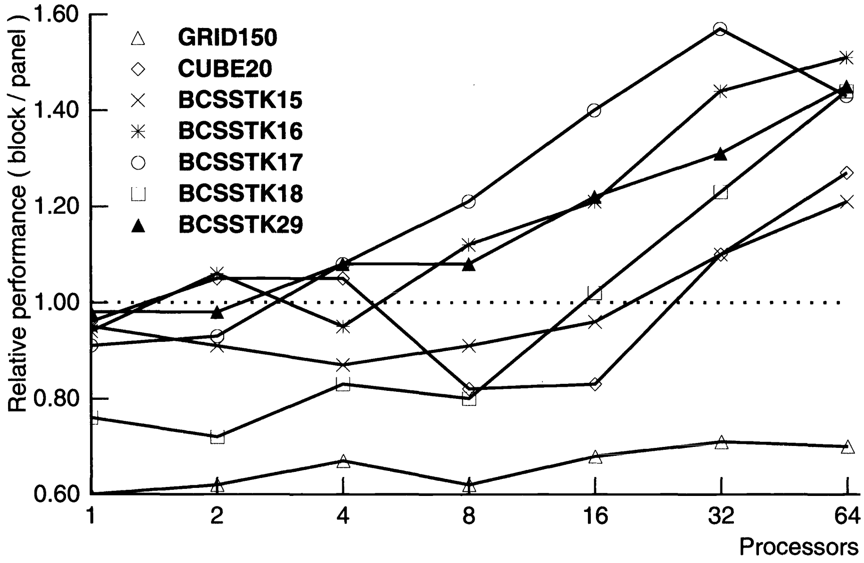


FIG. 4. *Relative performance (block fan-out versus panel multifrontal) on the iPSC/860 system.*

give the best performance obtained using a block size of 32, 48, or 64. We limit the number of cases we look at to limit the time required to gather performance data. For both methods, however, these partition size choices were observed to produce performance numbers that are within a few percent of the best that we obtained with any partition size. One exception is for large problems on a small number of nodes, where larger partition sizes that the ones we considered may give better performance.

The data in the figure indicates that performance of these methods is quite low, even on a large number of processors. Further investigation of the panel multifrontal method reveals that the main factor that limits performance is the critical path. As an example, consider problem BCSSTK15. If we assume a per-word communication cost of 55 floating-point operations (our estimate for the iPSC/860 system) and a panel width of 16 columns, then we find that the critical path for this problem limits parallel speedups to roughly 6-fold, no matter how many nodes are used. The corresponding bound for the larger BCSSTK29 is 7-fold speedup. Given these performance bounds, it is not surprising that achieved performance levels are low. If we look at the factors that contribute to this critical path, we find that communication costs account for more than half, and computation costs account for the rest. Recall that the amount of computation on the critical path can be reduced by reducing the panel width (at the cost of reduced single-node performance). Communication costs on the critical path, on the other hand, are unaffected by the panel width.

Now let us consider the performance of the block fan-out method on the iPSC/860 system. To allow a more direct comparison between panel multifrontal and block fan-out performance, Figure 4 shows the ratio of block fan-out performance to panel multifrontal performance for our benchmark problems. Note that performance is quite similar for a small number of nodes (with a few exceptions), and that performance diverges as the number of nodes increases. The block fan-out method produces significantly higher performance for 64 nodes.

Further investigation reveals that there are two primary performance bottlenecks for the block fan-out method on a large number of nodes. The first is load imbalance. As an example, for problem BCSSTK15 on 64 nodes using a block size of 48 (the block size that gives the best performance), the node with the most work assigned to it has 2.5 times more work than

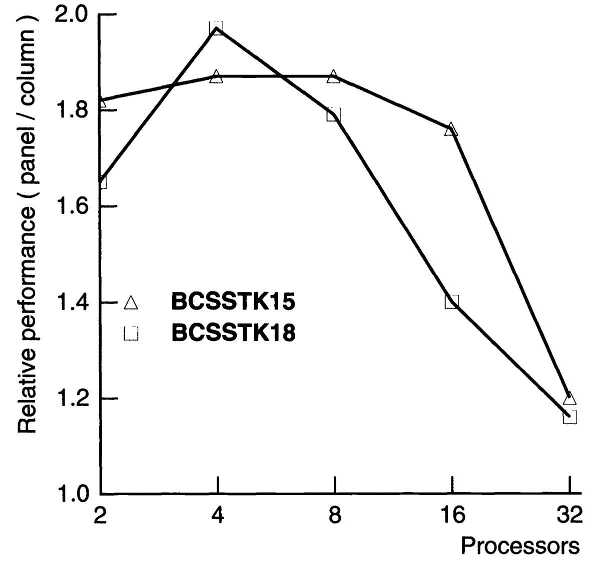


FIG. 5. *Relative performance (panel multifrontal versus column multifrontal) on the iPSC/860 system.*

it would with a perfect load balance. The second bottleneck is communication costs. The computation-to-communication ratio for our example is roughly 35 floating-point operations per communicated word. Recall that one word of communication costs roughly the same as 55 floating-point operations on the iPSC/860 system. This means that, on average, a node spends 1.6 times as much time communicating data as it spends performing computation. We find that the node that must perform 2.5 times as much work as it ideally would also must perform roughly 2.5 times as much communication as it ideally would. These two bottlenecks lead to a best-case parallel speedup for this problem of around 11-fold. The corresponding bound for BCSSTK29 is 12-fold. The critical path, which was quite constraining for the panel multifrontal method, is not very constraining for the block fan-out method. It limits parallel speedups to around 18-fold and 22-fold for BCSSTK15, and BCSSTK29, respectively. Overall, the performance upper bounds are much less constraining for the block fan-out method than they are for the panel multifrontal method. The result is higher performance on larger numbers of processors.

To evaluate the advantages of a panel or block approach to the computation over a more traditional column method, we now compare the performance numbers for our panel multifrontal method against the best performance numbers we are aware of for a column multifrontal method on the iPSC/860 system [21]. Figure 5 shows relative performance for two matrices where our studies overlap. Notice that the panel method gives roughly 80% higher performance for small numbers of nodes, but the performance difference decreases with increasing numbers of nodes. The reason is simply that the panel method improves per-node performance but increases the length of the critical path. Critical path costs are the more important determinant of performance when using more than a few nodes.

We therefore find that the block fan-out method provides higher performance than the panel multifrontal method on the iPSC/860 system, and both provide higher performance than a column approach. However, none of these methods can truly overcome the performance limitations that we observed on this machine.

4.2. Paragon. We now investigate factorization performance on the Paragon system. Figure 6 shows absolute parallel factorization performance on 1 to 64 nodes, and Figure 7 compares the performance of the Paragon system against that of the iPSC/860 system for both

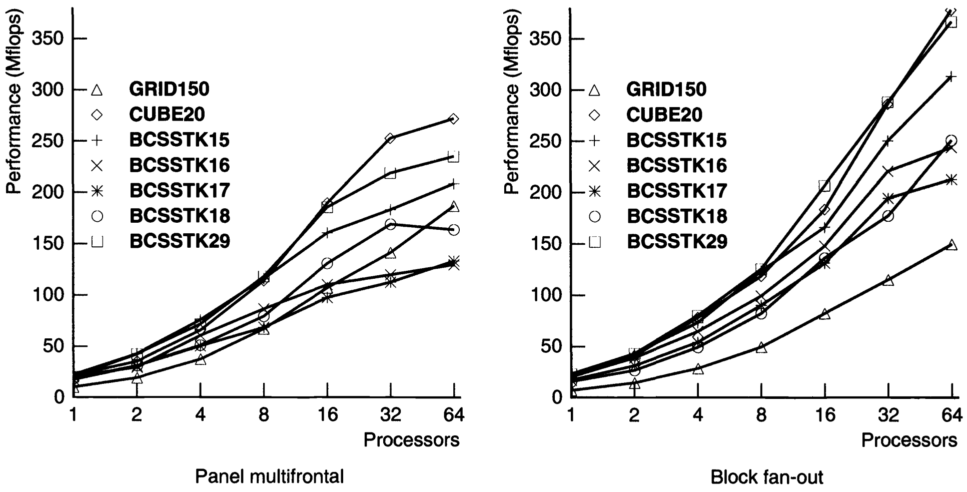


FIG. 6. *Factorization performance on the Paragon system.*

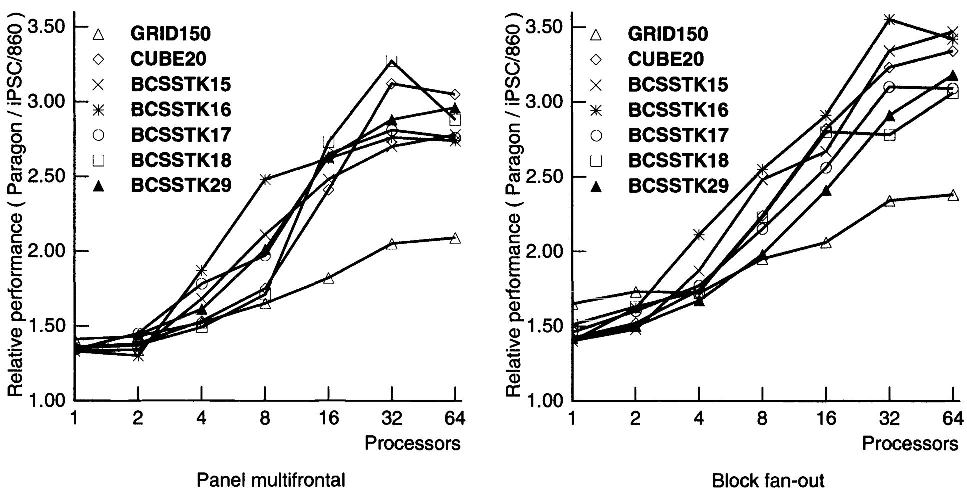


FIG. 7. *Performance improvement on the Paragon system (relative to the iPSC/860 system).*

the panel multifrontal and block fan-out methods. The figures show that the Paragon system provides substantially higher performance, and that the performance improvement grows as the number of nodes is increased. Improvements on 32 or 64 nodes are between 2.5-fold and 3.5-fold, depending on the matrix and the method.

These observed performance improvements can be better understood as follows. First, we note from the figures that the Paragon system provides higher single-node performance than the iPSC/860 system (roughly 40% higher for the panel multifrontal method and roughly 50% higher for the block fan-out method). Reasons include (1) a 25% increase in processor clock speed, (2) a factor of two increase in cache size (from 8 KBytes to 16 KBytes), and (3) the addition of a quad-word load instruction, which reduces the cost of fetching data from memory.

To better understand the performance increase for the panel multifrontal method, recall that on the iPSC/860 system the primary performance bottleneck for this method on a large number of processors was the critical path length. More than half of this critical path was

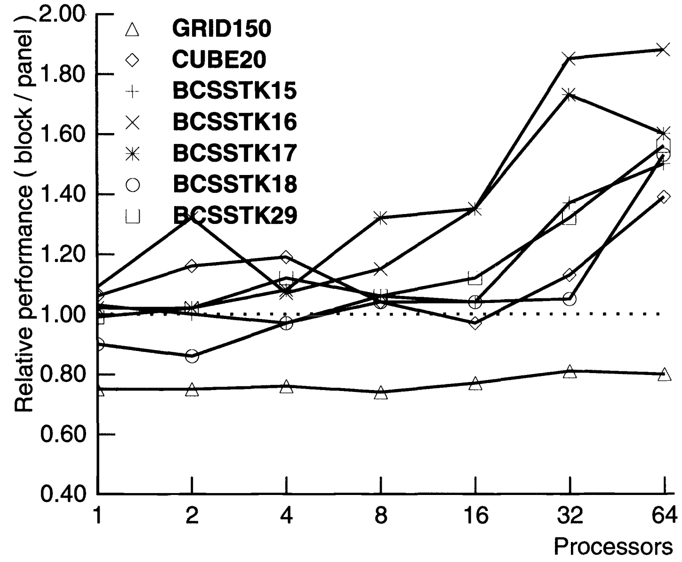


FIG. 8. *Relative performance (block fan-out method versus panel multifrontal) on the Paragon system.*

due to interprocessor communication costs. On the Paragon system, communication costs are reduced by a factor of more than 10. As a result, the overall critical path is roughly halved in length. Equivalently best-case parallel speedups are roughly doubled. When one combines this doubling in best-case speedup with a 40% improvement in single node performance, one finds that the improvement in observed performance is quite similar to the improvement in the performance bound.

For the block fan-out method, recall that the main performance bottleneck on the iPSC/860 system was the uneven distribution of work and communication among the nodes. The work distribution naturally has not changed. However, the communication costs, which magnified the performance impact of this imbalance by more than a factor of two, have been reduced by a factor of more than 10. The performance bound is therefore improved by roughly a factor of two. This factor of two improvement, combined with the 50% improvement in node performance, is consistent with the observed overall performance improvement.

Let us now compare the performance of the panel multifrontal and block fan-out methods on the Paragon system. Figure 8 shows relative performance numbers. This figure shows that the block fan-out method again produces significantly higher performance for larger numbers of nodes.

While parallel factorization performance is substantially higher on the Paragon system than it was on the iPSC/860 system, it is still not outstanding. As an example, we obtain roughly 300 Mflops for BCSSTK15 on 64 nodes, or roughly 5 Mflops per node. Considering that the sequential code produces nearly 25 Mflops on a single node, the nodes in the parallel method are clearly not being very well utilized. Our performance analysis indicates that the main performance bottleneck for the panel multifrontal method on the Paragon system is the cost of the floating-point operations on the critical path. The critical path length can be reduced by decreasing the panel width, but this leads to a drop in per-node performance. The decrease was observed to cancel the advantage of the shorter critical path. For the block fan-out method the main bottleneck is load imbalance. Again, this bottleneck can be improved by moving to a smaller block size, but the advantage was again observed to be cancelled by the resulting drop in per-node performance. The real solution in both cases is to consider larger problems.

TABLE 2
Larger benchmark matrices.

Problem name	Equations	NZ in A	NZ in L	Ops for factorization
BCSSTK31	35,588	1,181,416	5,485,320	2,550,990,053
COPTER2	55,476	759,952	13,556,728	11,377,269,959
CUBE30	27,000	183,600	6,980,825	3,904,329,794
CUBE35	42,875	292,775	13,347,541	10,114,682,189

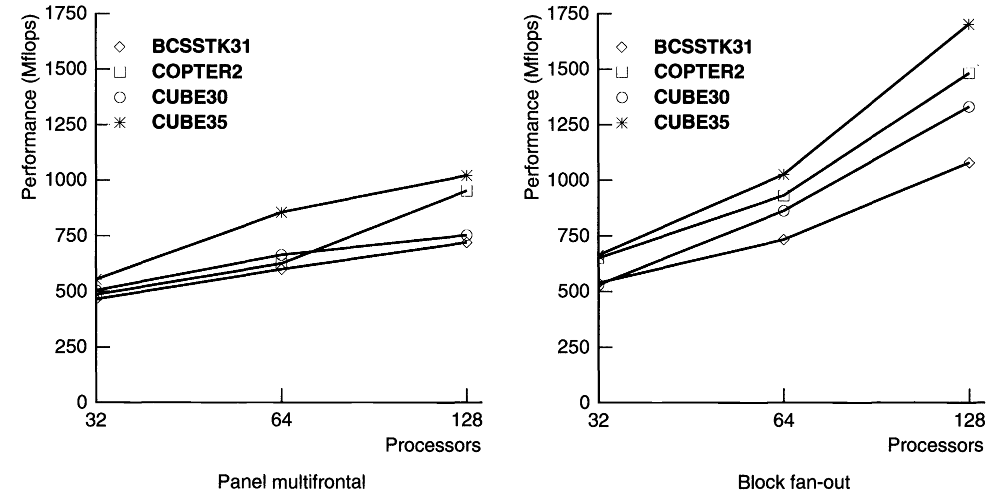


FIG. 9. *Paragon system performance for larger problems.*

5. Larger problems. It is our belief that the matrices we have considered so far, which are some of largest matrices in the Harwell-Boeing sparse-matrix test set, are not representative of the size of problems that people would want to solve on a large parallel machine. We believe that the primary reason these matrices have been used as benchmarks in the past is simply that larger matrices would not fit in the memories that were available on earlier machines. Fortunately, the larger amounts of memory available on newer parallel systems allow us to consider much larger problems. We now look at performance for matrices that are perhaps more appropriate factorization benchmark matrices for large parallel machines. Table 2 gives descriptions of these matrices. Problem BCSSTK31 is the largest problem in the Harwell-Boeing collection. Problem COPTER2 comes from a model of a helicopter rotor. It was given to us by Horst Simon at NASA Ames. The other two problems are simple 3-D grid problems. Note that these matrices are by no means unrealistically large. Even the largest one, COPTER2, requires only a few minutes to factor on a high-performance workstation (and a few seconds on a parallel machine).

Figure 9 shows factorization performance for these matrices on 32, 64, and 128 nodes of the Paragon system. Note that performance for these larger problems is extremely high. We obtain as much as 1.7 Gflops on 128 nodes. Also note that the performance of the block fan-out method diverges from the performance of the panel multifrontal method with increasing numbers of nodes. On 128 nodes, the block fan-out method provides roughly 70% higher performance than the panel multifrontal method. Further analysis reveals that the performance of the panel multifrontal method is again constrained by the critical path for these larger problems on 128 nodes. The improved scalability of the block fan-out method appears to become extremely important for these problems when more than 64 nodes are used.

6. Discussion and future work. The results of this paper demonstrate that it is possible to obtain very high performance for sparse Cholesky factorization on a distributed-memory multiprocessor. Indeed, the Paragon system is quite competitive with vector supercomputers for this computation. The 650 Mflops performance of a 32-node Paragon system is more than twice the performance of one CRAY Y-MP processor [14, 20]. Similarly, the 1.7 Gflops performance of a 128-node Paragon system is more than five times the performance of a CRAY Y-MP processor.

An obvious next step is to try to obtain even higher performance by using a larger number of nodes. We believe that the block fan-out method is the more promising of the two methods we have considered for doing this. One reason is that the block fan-out method produces asymptotically shorter critical paths than the panel multifrontal method. The critical path limited panel multifrontal performance on 128 nodes, so we would expect it to severely limit performance when using more nodes. A second reason is that the block fan-out method requires asymptotically less communication than the panel multifrontal method. The growth rate difference did not lead to a significant difference in communication volume on 128 or fewer nodes, but the communication volume difference would increase with increasing P.

We should note that one important piece missing from the scalable sparse factorization puzzle is scalable preprocessing. For this study, we performed most of the preprocessing sequentially, often on a separate workstation. The performance of the numerical factorization code is now sufficiently high that the preprocessing phases can require as much time as the parallel factorization. We are currently investigating parallel approaches to heuristic reordering and symbolic factorization.

7. Conclusions. This paper has investigated the impact of new technologies on parallel sparse Cholesky factorization performance. Specifically, we considered the impact of new algorithms (the panel multifrontal and block fan-out methods) and new hardware (the Intel Paragon system).

On the algorithm front, we found that these new algorithms improve parallel performance substantially on small iPSC/860 systems, providing nearly a factor of two improvement over a column method on eight nodes or fewer. With more nodes, however, the improvements are much smaller. The main performance bottleneck for multifrontal methods on the iPSC/860 system, the cost of communicating data on the critical path, is not addressed by the panel multifrontal method. The block fan-out method relieves this bottleneck, and it produces significantly higher performance for large numbers of nodes. However, it still suffers from several important bottlenecks. Parallel performance on the iPSC/860 system is still relatively low.

On the hardware front, we find that the Intel Paragon system, which delivers an order of magnitude higher communication bandwidth than the earlier Intel iPSC/860 system, provides 2.5 to 3.5 times the performance of the iPSC/860 system for large numbers of nodes (32 or more) using traditional benchmark matrices. Furthermore, we find that by considering larger benchmark matrices, we obtain extremely high performance. The Paragon system performs the factorization at 650 Mflops on 32 nodes, 1 Gflop on 64 nodes, and 1.7 Gflops on 128 nodes.

We conclude from this investigation that the factors that have historically limited parallel sparse factorization performance have been overcome, and that distributed-memory machines can now deliver extremely high factorization performance. We believe that these machines are now quite appealing platforms for performing this important computation, and that they will become increasingly so in the future.

REFERENCES

[1] P. Amestoy and I. Duff, *Vectorization of a multiprocessor multifrontal code*, Internat. J. Supercomputer Appl., 3 (1989), pp. 41-59.

[2] C. C. Ashcraft, S. C. Eisenstat, J. L. Liu, and A. H. Sherman, *A Comparison of Three Column-Based Distributed Sparse Factorization Schemes*, Research Report YALEU/DCS/RR-810, Computer Science Department, Yale University, New Haven, CT, 1990.

[3] C. C. Ashcraft and R. G. Grimes, *The influence of relaxed supernode partitions on the multifrontal method*, ACM Trans. Math. Software, 15 (1989), pp. 291-309.

[4] C. C. Ashcraft, R. G. Grimes, J. G. Lewis, B. W. Peyton, and H. D. Simon, *Recent progress in sparse matrix methods for large linear systems*, Internat. J. Supercomputer Appl., 1 (1987), pp. 10-30.

[5] J. Dongarra, J. Du Croz, S. Hammarling, and I. Duff, *A set of level 3 basic linear algebra subprograms*, ACM Trans. Math. Software, 16 (1990), pp. 1-17.

[6] I. S. Duff, R. G. Grimes, and J. G. Lewis, *Sparse matrix test problems*, ACM Trans. Math. Software, 15 (1989), pp. 1-14.

[7] G. Fox, et al., *Solving Problems on Concurrent Processors: General Techniques and Regular Problems*, Prentice-Hall, Englewood Cliffs, NJ, 1988.

[8] A. George, M. Heath, J. Liu, and E. Ng, *Solution of Sparse Positive Definite Systems on a Hypercube*, Tech. Report TM-10865, Oak Ridge National Laboratory, Oak Ridge, TN, 1988.

[9] A. George and J. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[10] A. George, J. Liu, and E. Ng, *Communication results for parallel sparse Cholesky factorization on a hypercube*, Parallel Comput., 10 (1989), pp. 287-298.

[11] J. Gilbert and R. Schreiber, *Highly parallel sparse Cholesky factorization*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 1184-1197.

[12] J. Liu, *Modification of the minimum degree algorithm by multiple elimination*, ACM Trans. Math. Software, 12 (1986), pp. 127-148.

[13] R. Lucas, *Solving Planar Systems of Equations on Distributed-Memory Multiprocessors*, Ph.D. thesis, Stanford University, Stanford, CA, 1988.

[14] E. Ng and B. Peyton, *Block sparse Cholesky algorithms on advanced uniprocessor computers*, SIAM J. Sci. Comput., 14 (1993), pp. 1034-1056.

[15] A. Pothen and C. Sun, *A mapping algorithm for parallel sparse Cholesky factorization*, SIAM J. Sci. Comput., 14 (1986), pp. 1253-1257.

[16] E. Rothberg and A. Gupta, *An efficient block-oriented approach to parallel sparse Cholesky factorization*, in Proc. Supercomputing '93, Portland, OR, November 1993, pp. 503-512.

[17] ———, *An Evaluation of Left-Looking, Right-Looking, and Multifrontal Approaches to Sparse Cholesky Factorization on Hierarchical-Memory Machines*, Tech. Report STAN-CS-91-1377, Stanford University, Stanford, CA, 1991.

[18] E. Rothberg, *Exploiting the Memory Hierarchy in Sequential and Parallel Sparse Cholesky Factorization*, Ph.D. thesis, Stanford University, Stanford, CA, 1993.

[19] E. Rothberg and A. Gupta, *Techniques for improving the performance of sparse Cholesky factorization on multiprocessor workstations*, in Proc. Supercomputing '90, New York, NY, November 1990, pp. 232-243.

[20] H. Simon, P. Vu, and C. Yang, *Performance of a Supernodal General Sparse Solver on the CRAY Y-MP:* 1.68 *Gflops with Autotasking*, Tech. Report SCA-TR-117, Boeing Computer Services, Seattle, WA, 1989.

[21] C. Sun, *Efficient Parallel Solutions of Large Sparse SPD Systems on Distributed-Memory Multiprocessors*, Cornell Theory Center Technical Report CTC92TR102, Cornell University, Ithaca, NY, 1992.

SIAM J. SCI. COMPUT.
Vol. 17, No. 3, pp. 714-724, May 1996

010

A STABLE HIGH-ORDER INTERPOLATION SCHEME FOR SUPERCONVERGENT DATA*

STEVEN PRUESS† AND HONGSUNG JIN†

Abstract. A local collocation scheme is developed that yields stable high-order accurate interpolants of discrete data arising from the numerical solution of a differential equation. It should prove to be especially attractive for applications where data are superconvergent, e.g., spline collocation at Gauss points. For simplicity, the formulas are initially developed for a scalar equation, but generalizations are later given for systems. Numerical examples are shown that illustrate the stability, even for the case of highly nonuniform meshes which have proven difficult in prior studies.

Key words. stable interpolation, spline collocation, superconvergence, boundary value problems

AMS subject classifications. 65D05, 65L10, 65L60, 65L20

1. Introduction and mathematical background. Some difficulties encountered in attempting to uniformly preserve the superconvergent accuracy of meshpoint approximations to collocation solutions of two-point boundary value problems were discussed in Pruess [9]. While several of the piecewise polynomial interpolation schemes introduced there were adequate on many examples, all suffered in some particular cases, especially when highly nonuniform meshes were used. The essential difficulty is the lack of sufficient superconvergent data to produce high-order accurate uniform interpolants. To preserve the same order of accuracy, if data from neighboring intervals are used, this leads to inaccuracies when local mesh ratios are high (see [9] and [10]). In this paper we discuss and illustrate a means of overcoming this difficulty; because it appears essential to use only local data, the missing information is supplied by a local collocation of the underlying differential equation. The resulting algorithms are straightforward for linear differential equations; extensions to nonlinear problems by iteration on approximating linear problems should be self-evident.

Another alternative is to use an implicit Runge–Kutta approach. Cash [4] has developed such formulas for first-order systems; their derivation is tedious, especially for higher-order equations and for a variety of orders of accuracy. The collocation approach given here appears to be much more direct and general; the determination of which approach is more efficient will require comparison of specific implementations.

The remainder of this section presents the notation used and some mathematical background needed below. The next section contains material about Hermite-Birkhoff interpolation which is needed to understand the behavior of the algorithms to follow. Section 3 is concerned with developing the formulas for local collocation for a scalar equation; the stability of the algorithm also is established. These ideas are generalized to systems of differential equations in §4; the final section contains numerical examples.

While it is more practical to consider systems of differential equations, for simplicity we begin with a scalar linear differential equation:

$$D^m u = \sum_{j=1}^{m} c_j(x) D^{j-1} u + f(x) \tag{1.1}$$

*Received by the editors October 22, 1993; accepted for publication (in revised form) December 20, 1994. This research was partially supported by National Science Foundation grant DMS-9322615 to the Colorado School of Mines.

†Department of Mathematical and Computer Sciences, Colorado School of Mines, Golden, CO 80401-1887 (spruess@slate.mines.edu, hjin@mines.edu).

for $a < x < b$. Of course, m boundary conditions also are needed, but because they will not play a major role in this paper, their details are ignored. Here D denotes differentiation with respect to x. We assume that the coefficient functions in equation (1.1) are "sufficiently smooth" and that (1.1) and the boundary conditions are such that there is a unique solution $u(x)$ for each choice of $f(x)$.

All approximate solutions considered will be piecewise polynomials. The set of breakpoints (also called nodes or meshpoints) for the pieces is denoted by Δ. With $\Delta = \{a = x_1 < x_2 < \cdots < x_{N+1} = b\}$, set $h_n = x_{n+1} - x_n$ and $h = \max h_n$. The symbol $P_{k,\Delta}$ denotes the space of piecewise polynomials of order k (degree $< k$) with breakpoints in Δ.

If collocation at Gaussian points (e.g., see de Boor and Swartz [2]) is used to produce an approximation $\hat{u}(x)$ in $P_{m+k,\Delta} \cap C^{m-1}[a, b]$ to the exact solution $u(x)$, then for $j = 1, 2, \ldots, m$ the errors satisfy (when $k \geq m$)

$$|D^{j-1}u(x_n) - D^{j-1}\hat{u}(x_n)| \leq Ch^{2k} \tag{1.2}$$

at the breakpoints. At general points x not in Δ,

$$|D^{j-1}u(x) - D^{j-1}\hat{u}(x)| \leq Ch^{m+k-j} \tag{1.3}$$

for some generic constant C. Hence, superconvergence occurs as long as $k > m$. Superconvergent accuracy on a discrete set of points also occurs when Galerkin approximations are used; e.g., see Bramble and Schatz [3] or Zlamal [11].

In Pruess [9] several high-order interpolation schemes were derived in an attempt to preserve the superconvergent accuracy. While all were proven to have uniform $O(h^{2k})$ errors as $h \to 0$, as previously mentioned, the actual errors for typical meshes were not satisfactorily small. Mathematically, nearly all the methods presented had large constants (involving local mesh ratios) in the h^{2k} term in the error expansion. We view this as a case of "instability": the small errors in $\hat{u}$ at meshpoints were magnified by the interpolation schemes used. In §3 we present a family of methods which does not suffer from this difficulty; i.e., the uniform error in the interpolation scheme has the same $O(h^{2k})$ accuracy but the error constant does not contain mesh-dependent quantities.

2. A Hermite-Birkhoff interpolation problem. In this section we digress with a discussion of a particular Hermite-Birkhoff interpolation problem which will be used to explain the behavior of the algorithms developed in later sections. The problem is the following.

Given integers m and k with $k > m$, some sufficiently smooth function $f(x)$, and a set of points $\{\zeta_i\}$ in $[-1, 1]$, find a polynomial $p(x)$ of order $2k$ satisfying

1. $(D^{j-1}p)(-1) = (D^{j-1}f)(-1)$ for $j = 1, 2, \ldots, m$;
2. $(D^{j-1}p)(1) = (D^{j-1}f)(1)$ for $j = 1, 2, \ldots, m$;
3. $(D^m p)(\zeta_i) = (D^m f)(\zeta_i)$ for $i = 1, 2, \ldots, 2k - 2m$.

Of course, it is well known that, in general, there may be no solution or many solutions to Hermite-Birkhoff problems. Some references for theory on Hermite-Birkhoff interpolation are [5], [7], and [8]. A Hermite-Birkhoff problem is said to be *poised* if for zero data ($f \equiv 0$) the only solution is $p(x) \equiv 0$. In the thesis of Jin [6] it was shown that the stated problem (for typical k and m) is poised for most choices of points including the important ones (for the algorithms in §§3 and 4) of uniform, either open (not including ±1) or closed (including ±1). This was done by explicitly calculating the determinant of the coefficient matrix associated with the interpolation conditions using a monomial representation of $p(x)$. In contrast, it was noticed that the choice of Gauss points (zeros of Legendre polynomials) was not poised for most k and m. The latter can be easily shown without computing any determinants.

LEMMA 2.1. *Assume*

$$p \in P_{2k}; \tag{2.1}$$

$$D^{j-1}p(\pm 1) = 0 \textit{ for } j = 1, 2, \ldots, m; \tag{2.2}$$

$$D^m p(\zeta_i) = 0 \textit{ for some } \{\zeta_i\} \in [-1, 1]. \tag{2.3}$$

Then $q = D^m p$ satisfies

$$q \in P_{2k-m}, \tag{2.4}$$

$$q(\zeta_i) = 0, \tag{2.5}$$

$$\int_{-1}^{1} t^{j-1} q(t)\, dt = 0 \textit{ for } j = 1, 2, \ldots, m. \tag{2.6}$$

Moreover, if q satisfies (2.4)-(2.6) *then p, the mth antiderivative of q with $p^{(j-1)}(-1) = 0$ for $j = 1, 2, \ldots, m$, satisfies* (2.1)-(2.3).

Proof. The consequences (2.4) and (2.5) are trivial; (2.6) follows inductively using repeated integrations by parts. Conversely, if (2.4)-(2.6) hold, then (2.1), (2.3), and half of (2.2) follow trivially. To show that $p^{(j-1)}(1) = 0$ for $j = 1, 2, \ldots, m$, it is also straightforward using integration by parts inductively.

Consequently, to show the stated Hermite-Birkhoff problem is not poised we can either find a nonzero p satisfying (2.1)-(2.3) or a nonzero q satisfying (2.4)-(2.6).

THEOREM 2.2. *The stated Hermite-Birkhoff problem is not poised when the set $\{\zeta_i\}$ is the set of zeros of the $(2k - 2m)$th degree Legendre polynomial L_{2k-2m} and $3m \leq 2k$.*

Proof. Let $q = L_{2k-2m}$; then for $m \leq 2k - 2m$ the conditions (2.4)-(2.6) trivially hold.

The condition that $3m \leq 2k$ is usually satisfied for typical choices of m and k arising from two-point boundary value problems. In fact, because we are requiring $k > m$ for superconvergence, the condition always holds for first- and second-order differential equations.

3. Formulas and theory for a single equation. As in §1 we denote the superconvergent approximation to the solution of the differential equation by $\hat{u}(x)$. The final uniformly high-order accurate approximation will be written $\bar{u}(x)$. We assume that $\hat{u}$ has errors satisfying (1.2)-(1.3) with $k > m$.

The approximation $\bar{u}(x)$ is characterized by the following.

1. $\bar{u}(x)$ is a piecewise polynomial of order $2k$ with breakpoints in Δ.
2. $(D^{j-1}\bar{u})(x_n) = (D^{j-1}\hat{u})(x_n)$ for $1 \leq j \leq m$; $x_n \in \Delta$.
3. On each $[x_n, x_{n+1}]$,

$$D^m \bar{u} = \sum_{j=1}^{m} c_j D^{j-1}\bar{u} + f \tag{3.1}$$

is evaluated at some ξ_{in}.

Since $\hat{u}$ is a piecewise polynomial of order $2k$, there are a total of $2kN$ coefficients in a local representation. The interpolation conditions determine $2mN$ of these, leaving $2(k - m)$ to be determined by the collocation conditions (3.1). This says that there should be $2k - 2m$ of the ξ's for each subinterval. These points will be referred to as the set of *secondary* collocation points.

A local Hermite representation is used for $\bar{u}$; i.e., on $[x_n, x_{n+1}]$,

$$\begin{aligned} \bar{u}(x) = &\sum_{j=1}^{k} (D^{j-1}\bar{u})(x_n^+)\Phi_j\left(-1 + 2\frac{x_{n+1} - x}{h_n}\right)\left(\frac{-h_n}{2}\right)^{j-1} \\ &+ \sum_{j=1}^{k} (D^{j-1}\bar{u})(x_{n+1}^-)\Phi_j\left(-1 + 2\frac{x - x_n}{h_n}\right)\left(\frac{h_n}{2}\right)^{j-1} \end{aligned} \tag{3.2}$$

where $\Phi_j(t)$ satisfies

$$(D^{\nu-1}\Phi_j)(-1) = 0, \qquad 1 \le \nu \le k, \tag{3.3}$$

$$(D^{\nu-1}\Phi_j)(1) = \delta_{\nu j}, \qquad 1 \le \nu \le k, \tag{3.4}$$

with D here denoting differentiation with respect to t.

Because there are the same number of secondary collocation points on each interval, it is convenient to adopt the same relative spacing on each interval; i.e., write

$$\xi_{in} = x_n + (1+\zeta_i)\frac{h_n}{2} \tag{3.5}$$

for some choice of $\{\zeta_i\}$ in $[-1, 1]$ to be specified later. For simplicity, we additionally assume that the $\{\zeta_i\}$ are chosen symmetrically; i.e., for some

$$\eta_1 > \eta_2 > \cdots > \eta_{k-m} > 0,$$

we have $\zeta_i = -\eta_i$ for $i = 1, \dots, k-m$ and $\zeta_i = \eta_{2k-2m-i+1}$ for $i = k-m+1, \dots, 2k-2m$.

Then the secondary collocation equations (3.1) become

$$\begin{aligned}
&\sum_{j=1}^{k}(D^{j-1}\bar{u})(x_n^+)\Phi_j^{(m)}(-\zeta_i)\left(\frac{-h_n}{2}\right)^{j-m-1} + \sum_{j=1}^{k}(D^{j-1}\bar{u})(x_{n+1}^-)\Phi_j^{(m)}(\zeta_i)\left(\frac{h_n}{2}\right)^{j-m-1} \\
&\quad = \sum_{\nu=1}^{m} c_\nu(\xi_{in})\left\{\sum_{j=1}^{k}(D^{j-1}\bar{u})(x_n^+)\Phi_j^{(\nu-1)}(-\zeta_i)\left(\frac{-h_n}{2}\right)^{j-\nu}\right. \\
&\quad\quad \left. + \sum_{j=1}^{k}(D^{j-1}\bar{u})(x_{n+1}^-)\Phi_j^{(\nu-1)}(\zeta_i)\left(\frac{h_n}{2}\right)^{j-\nu}\right\} + f(\xi_{in})
\end{aligned}$$

for $1 \le i \le 2k-2m$. Rearranging this and using the fact that $\bar{u}$ interpolates $\hat{u}$ at the endpoints of each piece, we have

$$\begin{aligned}
&\sum_{j=m+1}^{k}(D^{j-1}\bar{u})(x_n^+)\left(-\frac{h_n}{2}\right)^{j-m-1}\left\{\Phi_j^{(m)}(-\zeta_i) - \sum_{\nu=1}^{m} c_\nu(\xi_{in})\Phi_j^{(\nu-1)}(-\zeta_i)\left(-\frac{h_n}{2}\right)^{m+1-\nu}\right\} \\
&+ \sum_{j=m+1}^{k}(D^{j-1}\bar{u})(x_{n+1}^-)\left(\frac{h_n}{2}\right)^{j-m-1}\left\{\Phi_j^{(m)}(\zeta_i) - \sum_{\nu=1}^{m} c_\nu(\xi_{in})\Phi_j^{(\nu-1)}(\zeta_i)\left(\frac{h_n}{2}\right)^{m+1-\nu}\right\} \\
&\quad = f(\xi_{in}) - \sum_{j=1}^{m}(D^{j-1}\hat{u})(x_n)\left\{\Phi_j^{(m)}(-\zeta_i)\left(\frac{-h_n}{2}\right)^{j-m-1}\right. \\
&\quad\quad \left. - \sum_{\nu=1}^{m} c_\nu(\xi_{in})\Phi_j^{(\nu-1)}(-\zeta_i)\left(\frac{-h_n}{2}\right)^{j-\nu}\right\} \\
&\quad\quad - \sum_{j=1}^{m}(D^{j-1}\hat{u})(x_{n+1})\left\{\Phi_j^{(m)}(\zeta_i)\left(\frac{h_n}{2}\right)^{j-m-1} - \sum_{\nu=1}^{m} c_\nu(\xi_{in})\Phi_j^{(\nu-1)}(\zeta_i)\left(\frac{h_n}{2}\right)^{j-\nu}\right\}.
\end{aligned} \tag{3.6}$$

For each subinterval this is a linear system of $2k-2m$ equations in the unknowns $(D^{j-1}\bar{u})(x_n^+)$ and $(D^{j-1}\bar{u})(x_{n+1}^-)$ for $m+1 \le j \le k$.

Note that the coefficient matrix on each subinterval is an $O(h)$ perturbation of the one for the corresponding (same k and m) Hermite-Birkhoff problem in the previous section. Hence, we would expect, for sufficiently fine grids, that the matrix is invertible for most choices of the η_i and typical k and m. On the other hand, we would anticipate near singularity (ill conditioning) if Gauss points are chosen for the secondary collocation points on problems where $3m \leq 2k$. It should also be pointed out that for singularly perturbed differential equations the $O(h)$ change may in fact be an $O(h/\epsilon)$ change for some small ϵ, and, consequently, the value of ϵ has a major effect on the conditioning of the linear system.

In Jin [6] it is proven that when the associated Hermite-Birkhoff problem is poised, then the projector associated with the local interpolation scheme is well defined (the above coefficient matrix is invertible) and is bounded independently of the mesh. From this it is straightforward to establish error bounds; the arguments follow those of [9] without the local change of variable (see [6] for details). Assuming the associated Hermite-Birkhoff problem is poised, the error has the expected form

$$|D^{j-1}u(x) - D^{j-1}\bar{u}(x)| \leq Ch^{2k-j+1} \tag{3.7}$$

for $j = 1, 2, \ldots, 2k$ and any x in $[a, b]$. The constant C can depend on $u(x)$ and its derivatives, k, m, and the choice of $\{\eta_i\}$, but not the mesh. Numerical experiments (some displayed in §5) confirm these rates of convergence.

4. Generalizations to mixed-order systems. As in [9] we consider the mixed-order system of linear equations

$$D^{m_l}u_l = \sum_{j=1}^{p}\sum_{i=1}^{m_j} c_{lij}D^{i-1}u_j + f_l, \quad 1 \leq l \leq p, \tag{4.1}$$

for x in $[a, b]$. Let

$$m^* = \sum_{j=1}^{p} m_j; \tag{4.2}$$

then m^* boundary conditions also are needed.

Analogous to the scalar case, we denote the lth component of the collocation approximation by $\hat{u}_l(x)$ and its interpolant by $\bar{u}_l(x)$. Then $\bar{u}_l(x)$ satisfies (for $1 \leq l \leq p$) the following:

1. $\bar{u}_l(x)$ is a piecewise polynomial of order $2k$ with breakpoints in Δ.
2. $(D^{j-1}\bar{u}_l)(x_n) = (D^{j-1}\hat{u}_l)(x_n)$ for $1 \leq j \leq m_l$; $x_n \in \Delta$.
3. On each $[x_n, x_{n+1}]$,

$$D^{m_l}\bar{u}_l = \sum_{\mu=1}^{p}\sum_{\nu=1}^{m_\mu} c_{l\nu\mu}D^{\nu-1}\bar{u}_\mu + f_l \tag{4.3}$$

is evaluated over some finite set of secondary collocation points.

Because each component is a piecewise polynomial of order $2k$, there are a total of $2kpN$ coefficients in a local representation. The interpolation conditions determine $2m^*N$ of these leaving

$$2[(k - m_1) + (k - m_2) + \cdots + (k - m_p)]N$$

to be determined by the collocation conditions (4.3). This says that there should be $2k - 2m_l$ secondary collocation points for each l and n. These points are denoted by ξ_{inl} and, analogous to (3.5), for convenience we require that there be $\{\zeta_{il}\}$ in $(-1, 1)$ such that

$$\xi_{inl} = x_n + (1 + \zeta_{il})\frac{h_n}{2}$$

for $i = 1, \dots, 2k - 2m_l$, $l = 1, \dots, p$, and $n = 1, \dots, N$.

As in §2 a local Hermite representation is used for each $\bar{u}_l$, i.e., on $[x_n, x_{n+1}]$

$$\begin{aligned} \bar{u}_l(x) = {} & \sum_{j=1}^{k} (D^{j-1}\bar{u}_l)(x_n^+)\Phi_j\left(-1 + 2\frac{x_{n+1} - x}{h_n}\right)\left(\frac{-h_n}{2}\right)^{j-1} \\ & + \sum_{j=1}^{k} (D^{j-1}\bar{u}_l)(x_{n+1}^-)\Phi_j\left(-1 + 2\frac{x - x_n}{h_n}\right)\left(\frac{h_n}{2}\right)^{j-1} \end{aligned} \tag{4.4}$$

where $\Phi_j(t)$ satisfies (3.3)-(3.4).

Then, after some rearrangement, (4.3) becomes

$$\begin{aligned} & \sum_{j=m_l+1}^{k} (D^{j-1}\bar{u}_l)(x_n^+)\left(\frac{-h_n}{2}\right)^{j-1-m_l} (D^{m_l}\Phi_j)(-\zeta_{il}) \\ & - \sum_{\mu=1}^{p} \sum_{j=m_\mu+1}^{k} (D^{j-1}\bar{u}_\mu)(x_n^+) \sum_{\nu=1}^{m_\mu} c_{l\nu\mu}(\xi_{inl})\left(\frac{-h_n}{2}\right)^{j-\nu} (D^{\nu-1}\Phi_j)(-\zeta_{il}) \\ & + \sum_{j=m_l+1}^{k} (D^{j-1}\bar{u}_l)(x_{n+1}^-)\left(\frac{h_n}{2}\right)^{j-1-m_l} (D^{m_l}\Phi_j)(\zeta_{il}) \\ & - \sum_{\mu=1}^{p} \sum_{j=m_\mu+1}^{k} (D^{j-1}\bar{u}_\mu)(x_{n+1}^-) \sum_{\nu=1}^{m_\mu} c_{l\nu\mu}(\xi_{inl})\left(\frac{h_n}{2}\right)^{j-\nu} (D^{\nu-1}\Phi_j)(\zeta_{il}) \\ & = f_l(\xi_{inl}) - \sum_{j=1}^{m_l}\left[(D^{j-1}\hat{u}_l)(x_n)\left(\frac{-h_n}{2}\right)^{j-1-m_l} (D^{m_l}\Phi_j)(-\zeta_{il})\right. \\ & \qquad \left. + (D^{j-1}\hat{u}_l)(x_{n+1})\left(\frac{h_n}{2}\right)^{j-1-m_l} (D^{m_l}\Phi_j)(\zeta_{il})\right] \\ & + \sum_{\mu=1}^{p}\sum_{\nu=1}^{m_\mu} c_{l\nu\mu}(\xi_{inl}) \sum_{j=1}^{m_\mu}\left[(D^{j-1}\hat{u}_\mu)(x_n)\left(\frac{-h_n}{2}\right)^{j-\nu} (D^{\nu-1}\Phi_j)(-\zeta_{il})\right. \\ & \qquad \left. + (D^{j-1}\hat{u}_\mu)(x_{n+1})\left(\frac{h_n}{2}\right)^{j-\nu} (D^{\nu-1}\Phi_j)(\zeta_{il})\right] \end{aligned} \tag{4.5}$$

in analogy with the scalar case (3.6). Hence, in an implementation this $2(kp - m^*)$-by-$2(kp - m^*)$ linear system must be solved for each subinterval.

5. Examples and conclusions. In this section the local collocation algorithm is tested to verify the expected behavior. Many examples were tried, but for brevity only four are presented here; the first three are those from [9]. The local interpolant $\bar{u}$ is compared with the collocation solution $\hat{u}(x)$. We do not repeat data for the other interpolants in [9] because overall, none of them was as effective as the $\bar{u}(x)$ introduced here.

Many different choices for the secondary collocation points also were explored. The ones considered follow:

1. open uniform: $\eta_i = (k - m + 1 - i)/(2k - 2m + 1)$,
2. closed uniform: $\eta_i = (2k - 2m - i)/(2k - 2m - 1)$,
3. Gauss points: $\{\eta_i\}$ is the set of roots of the Legendre polynomial of degree $2k - 2m$,
4. pseudo-Chebyshev points: $\eta_i = 1 - \sin[i\pi/(2k - 2m + 2)]$

for $1 \leq i \leq k - m$.

If we were ignorant of the material in §2, the choice of Gauss points would seem a natural one for collocation because of its importance in generating the $\hat{u}(x)$ which has superconvergent values at the breakpoints. Neither the open or closed uniform nor the pseudo-Chebyshev set should have nonpoised underlying Hermite-Birkhoff problems so their coefficient matrices should be better behaved. However, it was observed in [9] that interpolants using the differential equation at breakpoints (corresponding to $\zeta = -1$ or 1) frequently were poor on singularly perturbed problems because the errors $u^{(j-1)} - \hat{u}^{(j-1)}$ for $1 \leq j \leq m$ were magnified by the reciprocal of the perturbation parameter. Hence, we anticipate that, at least on singularly perturbed problems, the closed choice will be inferior to the open one. The motivation for the pseudo-Chebyshev set is to try a standard distribution which clusters points near the midpoint of the subintervals.

All numerical calculations in this section were done on an 80386-based PC using double precision arithmetic (approximately 16 decimal digits). In contrast to [9], we have not chosen to make operation counts in this work. Partly this is due to the difficulty of comparisons since k, m, p, and N are all parameters. There is no doubt that the interpolants using local collocation are more expensive to produce because they require additional (small) linear systems to be solved; however, their total effort is still minor compared with the time required to produce $\hat{u}(x)$.

In some of the tables below, condition numbers are displayed; in fact, these are estimates using the standard LINPACK algorithms. Because these estimates are felt to be reliable as to the order of magnitude, we have not bothered to compute exact condition numbers themselves. However, they must be interpreted with some caution; row and column scaling can have a considerable affect on the condition number estimate even though the answer to the associated linear system was usually unaffected. Also, rather than implement (4.5) as given, we have actually used an equivalent system of equations where each (x_n, x_{n+1}) was mapped into $[0, 1]$ rather than $[-1, 1]$.

The first example is the first from [9] which is a system of two ordinary differential equations.

$$u_1''' = u_1' - 4e^{2x}u_2 + 4(x^2 + 1)e^x,$$

$$u_2' = -xu_1 + xu_1'' - u_2 - 4x^2e^x + (2x - 1)e^{-x},$$

$$u_1(0) = u_2(0) = u_1(1) = u_2(1) = 0.$$

The exact answers are $u_1(x) = x(x - 1)\exp(x)$ and $u_2(x) = x(x - 1)\exp(-x)$. The errors (nodal and uniform) for the standard collocation approximation $\hat{u}(x)$, and for $\bar{u}(x)$ based on several choices of secondary collocation points are shown in Table 1. All cases used $k = 4$ and $N = 12$ equally spaced breakpoints (nodes). We have also tabulated the maximum over the mesh of a condition number estimate for the coefficient matrix of the secondary collocation

TABLE 1
Absolute errors for the first example.

	u_1	u_1'	u_1''	u_2	
At nodes	0.614E−12	0.301E−11	0.748E−11	0.936E−12	
Uniform					
$\hat{u}(x)$	0.138E−11	0.119E−09	0.125E−07	0.232E−08	
$\bar{u}(x)$					Max Cond
open	0.614E−12	0.568E−11	0.578E−09	0.483E−11	1.4E+06
closed	0.633E−12	0.329E−11	0.167E−09	0.285E−11	6.1E+05
Gauss	0.643E−12	0.431E−11	0.240E−09	0.137E−08	8.5E+12
Chebyshev	0.614E−12	0.307E−11	0.174E−09	0.244E−11	9.6E+06

equations. Note that the choice of Gauss points does produce the most ill conditioned of the coefficient matrices, although on this example the accuracy of $\bar{u}_1$ is relatively unaffected by this. The other three choices of secondary collocation points produce interpolants of comparable accuracy, all superior to the full collocation solutions $\hat{u}_1(x)$ and $\hat{u}_2(x)$. The decline in the derivative accuracy for $\hat{u}_1$ reflects the expected error bounds (3.7).

The next example also is from [1] and was studied in [9].

$$u'' = -xu'/\epsilon - (\pi x \sin^2 \pi x)/\epsilon - \pi^2 \cos \pi x,$$

$$u(-1) = 2, \qquad u(1) = 0,$$

for which

$$u(x) = \cos \pi x + \operatorname{erf}(x/\sqrt{2\epsilon})/\operatorname{erf}(1/\sqrt{2\epsilon}).$$

This problem has an interior layer near the origin. For a given mesh the *maximum local mesh ratio* is defined by

$$\max_i\{\max[h_i/h_{i-1}, h_{i-1}/h_i]\};$$

the *global mesh ratio* is

$$\max h_i / \min h_i.$$

For the initial choice of $\epsilon = 10^{-4}$ we used 36 breakpoints chosen so that an approximation to $||u^{(6)}||^{1/6}$ was roughly equidistributed. The maximum local mesh ratio was 3.0 while the global mesh ratio was 32.8; again $k = 4$ was used. A second case $\epsilon = 10^{-6}$ was used with 64 breakpoints based on the same equidistribution principle; here the maximum local mesh ratio was 9.8 while the global mesh ratio was 295.

The uniform errors in $\bar{u}(x)$ and $\bar{u}'(x)$ were better than those for standard collocation; in fact, the superconvergent accuracy in the derivative approximation did not degrade at all. Table 2 contains the data comparing $\bar{u}(x)$ with $\hat{u}(x)$. Again, the choice of Gauss points led to ill-conditioned matrices and inferior accuracies, especially in $\bar{u}'(x)$. The choice of closed points (including the nodes) for the secondary collocation points was poor as we had anticipated for singularly perturbed differential equations in the initial discussion of this section.

The third example is a generalization of the final example cited in [9], motivated by an example in [1]. This is a set of two second-order ODEs with boundary layers.

$$u_1'' = (u_1 - u_2)/\epsilon - u_1',$$

$$u_2'' = (u_2 - u_1)/\epsilon^2 - u_2',$$

TABLE 2
Absolute errors for the second example.

	$\epsilon = 10^{-4}$			$\epsilon = 10^{-6}$		
	u	u'		u	u'	
At nodes	0.140E−7	0.786E−5		0.741E−9	0.475E−5	
Uniform						
$\hat{u}(x)$	0.110E−6	0.213E−3		0.147E−7	0.219E−3	
$\bar{u}(x)$			Max Cond			Max Cond
open	0.327E−7	0.786E−5	1.6E+4	0.521E−8	0.475E−5	2.2E+06
closed	0.872E−5	0.276E−3	2.8E+4	0.183E−3	0.110E−1	2.8E+06
Gauss	0.110E−6	0.213E−3	4.8E+9	0.880E−7	0.220E−3	1.8E+12
Chebyshev	0.331E−7	0.786E−5	5.6E+4	0.587E−8	0.475E−5	8.7E+05

$$\begin{aligned} u_1(0) + u_1'(0) &= 3 + 2(1+\epsilon)\exp(-5/\epsilon), \\ u_2(0) &= 1 - 2\exp[-5(1+1/\epsilon)], \\ u_1(5) - u_2(5) &= 2(1+\epsilon)(1 - \exp[-5(1+1/\epsilon)]), \\ u_1'(5) + u_2(5) &= 1 + 2(2+\epsilon)\exp[-5(1+1/\epsilon)]. \end{aligned}$$

The exact answers are

$$\begin{aligned} u_1(x) &= 1 - 2\exp(-x) + 2\epsilon\{\exp[(x-5)/\epsilon] - \exp[-(1+1/\epsilon)x]\}, \\ u_2(x) &= 1 - 2\exp(-x) - 2\{\exp[(x-5)/\epsilon] - \exp[-(1+1/\epsilon)x]\}. \end{aligned}$$

This was tried for $k = 4$ and two values of ϵ; in each case, meshes were chosen to equidistribute an approximation of $||u^{(6)}||^{1/6}$. For $\epsilon = 10^{-2}$ the mesh consisted of 20 breakpoints with a local mesh ratio of 13.5 and a global mesh ratio of 178.3; when $\epsilon = 10^{-4}$ and $N = 28$ the mesh ratios were 466 and 18771, respectively. Table 3 displays the errors for the interpolant based on local collocation with several choices of secondary collocation points. The superiority of $\bar{u}_1(x)$ and $\bar{u}_2(x)$ from local collocation is quite evident. The best choice for secondary collocation points was the open uniform one though the pseudo-Chebyshev choice is only slightly inferior.

The final example is a scalar third-order one whose solution is mildly oscillatory.

$$\begin{gathered} u''' = -[28^2\pi^2/(1+7x)^4]u + [28^3\pi^2/(1+7x)^5]u', \\ u(0) = 1, \qquad u(1) = 0, \qquad u''(1) = 0. \end{gathered}$$

The exact answer is $u(x) = (1+7x)\cos[4\pi/(1+7x)]$. The output for different k and N values is found in Table 4. In both cases the mesh was chosen to equidistribute an approximation of $||u^{(k+3)}||^{1/(k+3)}$. When $k = 4$ for this example, we have $3m > 2k$ so the theory in §2 says nothing about the poisedness of the Hermite-Birkhoff problem associated with the choice of Gauss points. The arguments in Jin [6] show that it is in fact poised. In contrast, when $k = 5$ the Gauss points definitely do not have a poised Hermite-Birkhoff problem since now $3m \leq 2k$. The data in Table 4 are consistent with this.

In conclusion, it is clear that an interpolant based on local collocation using an open uniform set of secondary collocation points can do a very good job of preserving the superconvergent accuracy at breakpoints of $\hat{u}(x)$ and its first $m - 1$ derivatives. Even on very irregular meshes the stability of $\bar{u}$ is quite evident: for singularly perturbed differential equations, $\bar{u}(x)$ preserves the superconvergent accuracy of $\hat{u}(x)$ at the grid points as long as a good initial mesh has been found.

The choice of the "best" or at least a "better" choice of local collocation points merits further theoretical study; this is the subject of continuing research on the part of the authors.

TABLE 3
Absolute errors for the third example.

$\epsilon = 0.01$

	u_1	u_1'	u_2	u_2'	
At nodes	0.113E−7	0.168E−6	0.213E−6	0.193E−4	
Uniform					
$\hat{u}(x)$	0.123E−5	0.135E−4	0.122E−5	0.135E−2	
$\bar{u}(x)$					Max Cond
open	0.131E−7	0.168E−6	0.213E−6	0.193E−4	1.0E+05
closed	0.296E−7	0.443E−6	0.222E−5	0.417E−4	4.6E+04
Gauss	0.582E+0	0.881E+2	0.582E+0	0.881E+2	2.9E+20
Chebyshev	0.153E−7	0.168E−6	0.213E−6	0.193E−4	4.6E+05

$\epsilon = 0.0001$

	u_1	u_1'	u_2	u_2'	
At nodes	0.752E−9	0.541E−7	0.606E−7	0.566E−3	
Uniform					
$\hat{u}(x)$	0.208E−6	0.309E−5	0.209E−6	0.309E−1	
$\bar{u}(x)$					Max Cond
open	0.936E−9	0.541E−7	0.606E−7	0.566E−3	9.3E+08
closed	0.300E−6	0.252E−5	0.301E−2	0.253E−1	3.5E+08
Gauss	0.203E+4	0.231E+5	0.203E+4	0.231E+5	3.9E+23
Chebyshev	0.114E−8	0.541E−7	0.606E−7	0.566E−3	3.5E+09

TABLE 4
Absolute errors for the fourth example.

	$k = 4$	$N = 24$		$k = 5$	$N = 20$	
	u	u'	u''	u	u'	u''
At nodes	0.158E−7	0.863E−6	0.512E−4	0.323E−9	0.104E−7	0.376E−6
Uniform						
$\hat{u}(x)$	0.826E−7	0.443E−4	0.328E−1	0.488E−8	0.401E−5	0.462E−2
$\bar{u}(x)$						
open	0.158E−7	0.538E−5	0.316E−2	0.323E−9	0.324E−7	0.225E−4
closed	0.478E−7	0.918E−5	0.555E−2	0.792E−7	0.458E−7	0.343E−4
Gauss	0.245E−7	0.434E−5	0.263E−2	0.657E+1	0.655E+3	0.548E+6
Chebyshev	0.173E−7	0.634E−5	0.325E−2	0.348E−9	0.503E−7	0.241E−4

Acknowledgment. We thank the referee for the careful reading of the manuscript, the remarks for improving the clarity of the presentation, and especially for the suggestions which led to a total revision of §2.

REFERENCES

[1] U. ASCHER, J. CHRISTIANSEN, AND R. RUSSELL, *A collocation solver for mixed order systems of boundary value problems*, Math. Comp., 33 (1979), pp. 659-679.

[2] C. DE BOOR AND B. SWARTZ, *Collocation at Gaussian points*, SIAM J. Numer. Anal., 10 (1973), pp. 582-606.

[3] J. BRAMBLE AND J. SCHATZ, *Higher order local accuracy by averaging in the finite element method*, Math. Comp., 31 (1977), pp. 94-111.

[4] J. CASH, *Continuous extensions of a deferred correction scheme for solving two-point boundary value problems*, preprint, 1992.

[5] D. FERGUSON, *The question of uniqueness for G. D. Birkhoff interpolation problems*, J. Approx. Theory, 2 (1969), pp. 1-28.

[6] H. JIN, *A Stable High Order Interpolation Scheme for Superconvergent Data*, M.S. Thesis, Colorado School of Mines, Golden, CO, 1993.

[7] S. KARLIN AND J. KARON, *On Hermite-Birkhoff interpolation*, J. Approx. Theory, 6 (1972), pp. 90-114.

[8] G. Lorentz, K. Jetter, and S. Riemenschneider, *Birkhoff interpolation*, in Encyclopedia of Mathematics and its Applications, Vol. 19, Addison-Wesley, Reading, MA, 1983.

[9] S. Pruess, *Interpolation schemes for collocation solutions of two point boundary value problems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 322-333.

[10] ———, *Stability bounds for local Lagrangian interpolation*, J. Approx. Theory, 53 (1988), pp. 117-127.

[11] M. Zlamal, *Some superconvergence results in the finite element method*, in Mathematical Aspects of Finite Element Methods, Springer-Verlag, Berlin, 1977, pp. 351-362.

SIAM J. SCI. COMPUT.
Vol. 17, No. 3, pp. 725–739, May 1996

AN ASSESSMENT OF NONMONOTONE LINESEARCH TECHNIQUES FOR UNCONSTRAINED OPTIMIZATION*

PHILIPPE L. TOINT†

Abstract. The purpose of this paper is to discuss the potential of nonmonotone techniques for enforcing convergence of unconstrained minimization algorithms from starting points distant from the solution. Linesearch-based algorithms are considered for both small and large problems, and extensive numerical experiments show that this potential is sometimes considerable. A new variant is introduced in order to limit some of the identified drawbacks of the existing techniques. This variant is again numerically tested and appears to be competitive. Finally, the impact of preconditioning on the considered methods is examined.

Key words. nonmonotone algorithms, linesearch, unconstrained optimization

AMS subject classifications. 90C30, 90C26, 49M37

1. Introduction.

In this paper, we consider algorithms for the numerical solution to the problem

$$\underset{x \in \mathbf{R}^n}{\text{minimize}} \quad f(x), \tag{1}$$

where $f(x)$ is a real-valued smooth function, which we assume is bounded below. Most widely available algorithms for solving the *nonlinear unconstrained minimization problem* (1) are, today, *descent methods*, in that they generate a sequence of iterates $\{x_k\}$ such that the associated sequence of objective function values $\{f(x_k)\}$ is monotonically decreasing. This is the case of all methods described in [9], a now classical reference on this subject, such as that of many packages for solving unconstrained problems, as well as UNCMIN [17], TENMIN [22], LANCELOT [7], and those from the Harwell Subroutines VA15 [18], VE08 [23], and VE10 [24], to cite just a few. This monotonicity property may be considered as natural, since one wishes to find a point with the lowest possible objective function value, and has been extensively used, both in theory and practice, to ensure that convergence to a minimizer occurs, even when the minimization is started far away from such a point (see [3], [4], [6], or [25] for examples of convergence theory heavily relying on this important property).

It has been noted, however, that imposing monotonicity of the sequence of objective function values is not without drawbacks. A typical case where monotonicity can cause severe loss of efficiency is when the objective function features deep narrow curved valleys: once trapped near the bottom of such a valley, the sequence of iterates must follow the valley's floor rather closely, which can result in very short steps or even undesired zigzagging. This observation has led researchers to propose algorithms that do not always guarantee the monotonicity properties. In particular, Grippo, Lampariello, and Lucidi published a few papers (see [13], [14], and [15]) where they demonstrated on some examples that substantial efficiency gains could be achieved in the solution of (1) if one is ready to abandon, at least to some extent, the constraint of generating monotonically decreasing sequences of objective function values. They also provided the convergence analysis to support their algorithmic proposals. In all fairness, Grippo, Lampariello, and Lucidi were not the only researchers interested in nonmonotone techniques for nonlinear optimization: the watchdog technique of Chamberlain et al. [5] is also a strategy of this type, although intended for the solution of constrained problems.

*Received by the editors May 31, 1994; accepted for publication (in revised form) January 3, 1995.

†Department of Applied Mathematics, Facultes Universitaires Notre Dame de la Paix, Rue de Bruxelles, 61, B-5000 Namur, Belgium (pht@math.fundp.ac.be).

Surprisingly, and despite their obvious potential practical implications, the findings of Grippo, Lampariello, and Lucidi didn't lead to any substantial validation by other researchers, at least as far as the author is aware. These ideas have been sometimes used in the literature (see [16], for instance), but have not yet been incorporated in the mainstream research on nonlinear unconstrained minimization. It is the first purpose of this paper to re-examine their proposals and contribute to their evaluation. The second purpose is to present a variant of their original methods, also based on a succession of unidimensional (approximate) minimizations, usually known as "linesearches." The third and final aim of the paper is to examine the impact of preconditioning on the considered nonmonotone algorithms.

The paper is organized as follows. The original proposals of Grippo, Lampariello, and Lucidi are considered in §2, where a specific implementation of these proposals is compared with a more classical monotone linesearch algorithm on a large collection of both small and larger test problems from the CUTE collection [2]. A variant of these methods is introduced and tested in §3. Section 4 is devoted to the numerical investigation of the effect of preconditioning. Finally, we present some conclusions and perspectives in §5.

2. The algorithms of Grippo, Lampariello, and Lucidi.

2.1. The original proposals. The classical framework in which Grippo, Lampariello, and Lucidi discuss their proposals is that of a "gradient related" method using an Armijo linesearch. If we denote the usual Euclidean inner product and norm by $\langle \cdot, \cdot \rangle$ and $\| \cdot \|$, respectively, and define $g_k = \nabla f(x_k)$, we may describe this basic method as follows.

Basic Linesearch Algorithm (BLS)

0. Initialization. A starting point x_0 is given, as well as constants $\beta \in (0,1)$ and $\gamma \in (0,1)$. Set $k = 0$.

1. Test for convergence. Stop if the chosen convergence criteria are met.

2. Determination of a search direction. Find a direction d_k such that

$$\langle d_k, g_k \rangle \leq -\kappa_1 \|g_k\|^2 \tag{2}$$

and

$$\|d_k\| \leq \kappa_2 \|g_k\| \tag{3}$$

for some positive constants κ_1 and κ_2.

3. Linesearch. Determine the smallest integer j_k such that, if one defines

$$\alpha_k = \beta^{j_k}, \tag{4}$$

then

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \gamma \alpha_k \langle g_k, d_k \rangle. \tag{5}$$

Then, set

$$x_{k+1} = x_k + \alpha_k d_k. \tag{6}$$

4. Prepare for the next iteration. Increment k by one and go to step 1.

There are several possible ways to determine the step d_k in step 2 of this prototype algorithm, in order to ensure that conditions (2) and (3) are satisfied. Grippo, Lampariello, and Lucidi propose to use the "modified Cholesky" factorization technique of Gill and Murray [10]. Other techniques will be discussed in § 2.3, in relation to our numerical experiments.

For future reference, we will refer to the linesearch calculation of step 3 as a procedure for obtaining x_{k+1} from the *base point* x_k (which determines the gradient $g_k = \nabla f(x_k)$), the *search direction* d_k, and the *reference value* $f(x_k)$ (the first term of the right-hand side of (5)). More concisely, we will express this relationship by the statement

$$x_{k+1} = LS_{\beta,\gamma}(x_k, d_k, f(x_k)), \tag{7}$$

which is therefore an alternative equivalent specification of step 3.

It is now easy to describe the original idea of Grippo, Lampariello, and Lucidi [13]: instead of using $f(x_k)$ as the reference value in step 3, they propose to use an objective function value corresponding to a previous iterate. More precisely, they propose to define

$$r_k = \max_{j=0,\dots,p} f(x_{k-j}) \tag{8}$$

for some fixed nonnegative integer p and then to replace step 3 in the basic algorithm by

$$x_{k+1} = LS_{\beta,\gamma}(x_k, d_k, r_k) \tag{9}$$

instead of (7). We will call the resulting algorithm the *GLL*1 *Algorithm*. As a consequence of (8), the strict monotonicity of the sequence $\{f(x_k)\}$ is no longer enforced, giving the algorithm more flexibility in the way in which successive iterates are obtained. Grippo, Lampariello, and Lucidi in fact describe and analyze several possible rules for the definition of the reference value r_k, but only provide numerical examples for the choice (8).

In a subsequent paper, Grippo, Lampariello, and Lucidi [15] introduce a further relaxation of the monotonicity property. Briefly speaking, they argue that some steps could be *automatically* accepted, without even checking any condition like (5), provided that these steps are sufficiently short (they require the subsequence of the norms of these steps to be geometrically converging to zero). To avoid convergence to undesirable maxima or saddle points, the value of the objective function is occasionally checked: if a suitable decrease has not been obtained since the last such verification, the algorithm is then restarted from the iterate corresponding to this last acceptable value. The precise manner in which the reference value r_k is redefined and this idea realized is a little more complex than the modification leading to Algorithm GLL1. It can be cast in the framework of Algorithm BLS as follows.

ALGORITHM GLL2

0. Initialization. A starting point x_0 is given, as well as constants $\beta \in (0,1), \gamma \in (0,1)$, $\eta \in (0,1)$, $\mu > 0$, $\xi \in (0,1)$, $\Delta_0 > 0$, $p \geq 1$, and $N \geq 1$. Set $k = \ell = 0$ and $f_0 = r_0 = f(x_0)$.

1. Test for convergence. Compute g_k and stop if $\|g_k\|$ is sufficiently small.

2. Determination of a search direction. Find d_k such that (2) and (3) hold and such that

$$\|d_k\| \geq \kappa_3 \|g_k\| \tag{10}$$

for some positive constant $\kappa_3 \leq \kappa_2$. If x_k has been obtained by step 5 or if $k = 0$, define $d_\ell = d_k$. Then, if $k = \ell + N$ go to step 3. Otherwise, go to step 4.

3. Tests after N **automatic step acceptances.** Compute $f(x_k)$ if necessary. If $f(x_k) \geq r_k$, set

$$x_k = x_\ell, \quad d_k = d_\ell, \quad \Delta_{k+1} = \Delta_k \tag{11}$$

and go to step 5. Otherwise, increment ℓ by one and define

$$x_\ell = x_k, \quad d_\ell = d_k, \quad f_\ell = f(x_k), \text{ and } r_k = \max_{j=0,\dots,p} f_{\ell-j}. \tag{12}$$

Then, if $\|d_k\| \le \Delta_k$, set

$$x_{k+1} = x_k + d_k, \quad \Delta_{k+1} = \eta\Delta_k, \quad r_{k+1} = r_k \tag{13}$$

and go to step 6. Otherwise go to step 5.

4. Tests before N automatic step acceptances. If $\|d_k\| \le \Delta_k$, perform step 4.a. Otherwise, perform step 4.b.

Step 4.a. If $\|d_k\| > \|d_{k-1}\|$, compute $f(x_k + d_k)$ and $f(x_k)$ if necessary. Then test if $f(x_k + d_k) - f(x_0) \ge \mu(f(x_0) - f(x_k))$. If this latter inequality holds, set

$$x_k = x_\ell, \quad d_k = d_\ell, \quad \Delta_{k+1} = \xi\Delta_k \tag{14}$$

and go to step 5. In all other cases, set x_{k+1}, Δ_{k+1}, and r_{k+1} according to (13) and go to step 6.

Step 4.b. Compute $f(x_k)$ if necessary. Then, if $f(x_k) \ge r_k$, set x_k, d_k and Δ_{k+1} using (11) and go to step 5. Otherwise, increment ℓ by one, recompute x_ℓ, d_ℓ, f_ℓ, and r_k using (12) and go to step 5.

5. Linesearch. Define

$$x_{k+1} = LS_{\beta,\gamma}(x_k, d_k, r_k), \tag{15}$$

increment ℓ by one and define

$$x_\ell = x_{k+1}, \quad f_\ell = f(x_{k+1}) \text{ and } r_{k+1} = \max_{j=0,\dots,p} f_{\ell-j}. \tag{16}$$

6. Prepare for the next iteration. Increment k by one and go to step 1.

We now briefly comment on this algorithm.

1. Condition (10) is introduced by Grippo, Lampariello, and Lucidi [15] and makes the length of the step an adequate measure of criticality. Section 2.3 shows that it does not constitute a strong additional requirement in a practical algorithm.

2. The step d_k is automatically accepted (see step 4) if its norm is at most equal to Δ_k and has not increased compared with that of d_{k-1}. In this case, the function value is not even computed. Observe also that the index ℓ denotes the last iterate at which the value of the objective function has been evaluated and has decreased enough. As indicated above, the algorithm "restarts" from this point if the objective function has not sufficiently diminished after N automatic acceptances.

3. The idea behind step 4.a is that it might be advisable to enforce closer control of the monotonicity of the objective function values if these become significantly larger than the initial value $f(x_0)$: this behavior combined with large steps may indeed indicate that the iterates are leaving the region of interest.

4. The mechanism for updating the reference value r_k is essentially identical to that used in GLL1, except that it only uses the last p function values that have been computed and which do not exceed their corresponding reference values. This distinction is necessary because, as indicated in the previous paragraph, some function values may not have been calculated when short steps are automatically accepted.

Grippo, Lampariello, and Lucidi [15] indicate that Algorithm GLL2 may be more efficient than GLL1 and often compare favorably with the modified Newton routine E04LBF from the NAG library.

2.2. Convergence theory. Grippo, Lampariello, and Lucidi proved that every limit point of the sequence of iterates produced by Algorithms GLL1 and GLL2 is a stationary point for problem (1). In order to obtain this result, they assume that the objective function is twice continuously differentiable on the compact level set $\{x \mid f(x) \leq f(x_0)\}$. Their argument is no longer based on the (abandoned) decreasing nature of the sequence of objective function values $f(x_k)$, but rather on the monotonicity of the references values r_k. The analysis is slightly more involved for Algorithm GLL2, since the effect of restarting the algorithm after an "unsuccessful" series of N automatically accepted steps must be taken into account; see [13] and [15] for details.

Note that the same convergence result is automatically obtained for Algorithm BLS by viewing this algorithm as a special case of Algorithm GLL1 where $p = 0$, but may also be obtained by directly exploiting (2)-(6). In particular, (2) and (3) imply that BLS produces "gradient-related" search directions (see [19, §14.3]).

2.3. An implementation. Our implementation of Algorithms BLS, GLL1, and GLL2 in Fortran 77 is rather straightforward.

- Termination occurs when $\|g_k\| \leq 10^{-5}$.
- The determination of the search direction d_k at step 2 is achieved by approximately solving the Newton system

$$(H_k + M_k)d_k = -g_k \tag{17}$$

where H_k denotes $\nabla^2 f(x_k)$, and where M_k is a symmetric positive semidefinite perturbation matrix such that the eigenvalues of $H_k + M_k$ are in the interval $[\kappa_4, \kappa_5]$ (for some positive constants κ_4 and κ_5 independent of k). The main motivation for allowing an approximate solution of (17) is the considerable reduction of computational effort (compared with a full solution) that is typically obtained for large-scale problems. The step d_k obtained as an approximate solution of (17) can thus be regarded as the exact solution of a "perturbed" Newton system of the form

$$(H_k + M_k)d_k = -g_k + h_k \tag{18}$$

where we request the "residual" h_k to satisfy the conditions

$$\langle d_k, h_k \rangle = 0 \tag{19}$$

and

$$\|h_k\| \leq \|g_k\| \min(0.1, \|g_k\|^{1/2}). \tag{20}$$

Among the many possible ways of selecting the perturbation matrix M_k, we have chosen the modified conjugate-gradients (CG) method proposed by Arioli et al. [1]. This method guarantees both (19) and the uniform positive definite and bounded character of the matrices $H_k + M_k$ by construction. Note that the modified Cholesky factorization technique used by Grippo, Lampariello, and Lucidi in their papers also guarantees the same conditions (with $h_k = 0$), as would other types of modified factorizations (see [7], [11], [21], or [20], for instance). However, since our approach allows for nonzero h_k, we have to enforce (20) explicitly: this is simply achieved in our code by stopping the modified CG method whenever this condition is satisfied. Note that condition (19) is automatically ensured by the properties of the CG algorithm.

It is now easy to verify that (2), (3), and (10) follow from (19), (20), and the requirements imposed on the matrices $H_k + M_k$. We note first that (20) and (18) imply that

$$\|(H_k + M_k)d_k\| \geq \|g_k\| - \|h_k\| \geq 0.9\|g_k\|, \tag{21}$$

which yields that

$$\|d_k\| \geq \frac{0.9}{\kappa_5}\|g_k\|, \tag{22}$$

and (10) is satisfied. Using now (19), one can prove (see [12]) that

$$\langle g_k, d_k\rangle \leq -[\mathrm{cond}(H_k + M_k)]^{-2}\|g_k\| \, \|d_k\| \tag{23}$$

where cond(X) is the ℓ_2 condition number of the matrix X. Since the eigenvalues of $H_k + M_k$ are contained in $[\kappa_4, \kappa_5]$, its condition number is bounded above by κ_5/κ_4, which, together with (10), gives (2). For the same reason, one immediately obtains that (3) holds. Indeed, using (18),

$$\|d_k\| \leq \frac{1}{\kappa_4}(\|g_k\| + \|h_k\|) \leq \frac{1.1}{\kappa_4}\|g_k\|, \tag{24}$$

where we used (20) to derive the second inequality.

- In our code, the constants $\mu = 10^6$ and $\xi = 10^{-3}$ are used in step 4.a, as recommended in [15].
- After some experimenting, a value of $p = 10$ was chosen in (12) and (16), the initial step threshold Δ_0 was set to unity, and the maximum number of successive automatic step acceptances N was fixed at 10.
- The linesearch procedures follows step 3 of Algorithm BLS nearly exactly (with $\gamma = 0.01$ and $\beta = 0.5$), except that doubling the stepsize is attempted if the unit step satisfies (5) and M_k nonzero, in an attempt to make a step as big as possible along a direction of (possibly) negative curvature.
- A maximum number of 10000 iterations and a maximum CPU time of five hours were allowed.

2.4. Numerical comparisons. We now present the results of an extensive numerical investigation of Algorithms BLS, GLL1, and GLL2. We have run these three methods on a set of 327 unconstrained problems available[1] from the CUTE collection [2]. All tests were performed on a DEC 3000/500 under OSF and using version 3.3 of the Fortran compiler with the default optimizing option. As the total number of test runs discussed in this paper exceeds 2500, it is impossible to include their detailed results in this text and we only present statistical summaries. The complete results, however, are available as a separate report [26].

In order to clarify the comparison, we divided our set of 327 problems into two subsets: the first contains all problems of dimension at most 500 (226 problems), and the second contains all larger cases (101 problems). This allows us to distinguish the effects of the nonmonotone strategies for small and large problems.

We first consider the relative reliability of our three algorithms on both problem sets and note that BLS failed on eleven small and six large problems, while GLL1 failed on nine small and five large problems and GLL2 on ten small and five large ones. Failures were caused by an excessive number of iterations, excessive CPU time, or by the occurrence of an arithmetic error during the computation of the objective function. The latter occurred often in the early iterations where a large step was attempted that lead outside the region where the problem functions are well defined on the computer.

Among the remaining 212 small and 93 large problems that could be solved by all three algorithms, six small ones and one large problem exhibited more than one local minimizer and, therefore, were rejected from our test sets, leaving 206 small and 92 large problems for further

[1] In March 1994.

TABLE 1
Cumulative statistics for BLS, GLL1, and GLL2.

Method	206 small problems				92 large problems			
	iters	evals	CG-its	time	iters	evals	CG-its	time
BLS	8173	17926	134099	1752	5041	9231	541429	35233
GLL1	8249	22793	153943	1288	7564	17135	966247	22661
GLL2	8089	21637	151501	1300	7504	16795	968919	23586

TABLE 2
Rankings for BLS, GLL1, and GLL2 on 206 *small and* 92 *large problems.*

Performance measure	Method	Small problems			Large problems		
		1st	2nd	3rd	1st	2nd	3rd
Iterations	BLS	174	20	12	80	7	5
	GLL1	153	51	2	67	20	5
	GLL2	162	39	5	68	20	4
Evaluations	BLS	57	110	39	30	48	14
	GLL1	41	140	25	24	56	12
	GLL2	166	37	3	69	19	4
CG iterations	BLS	160	30	16	69	14	9
	GLL1	156	45	5	72	15	5
	GLL2	166	36	4	77	12	3
Time	BLS	196	9	1	72	13	7
	GLL1	197	9	0	74	15	3
	GLL2	199	7	0	80	11	1

analysis. Table 1 shows the cumulative statistics for the three methods on these problems. In this table, "iters" stands for the total number of iterations needed to solve all problems, "evals" for the total number of objective function evaluations, "CG-its" for the total number of iterations within the modified conjugate-gradients procedure defining the search direction and "time" for the total CPU time (rounded to the second). Note that the objective function's gradient and Hessian are evaluated once per iteration.

These cumulative measures tend to suggest that GLL1 and GLL2 provide an average improvement in CPU time on BLS for small problems. This is obtained at the expense of more CG iterations and function evaluations. The situation is similar for the larger problems.

In order to refine our conclusions, we now look at more disaggregate performance measures and report, in Table 2, how many times each of the three considered methods has been best, second, or last. In these comparisons, two CPU times are considered equal if they correspond to runs yielding identical iterates or if they differ by at most 5% of the best or by at most half a second. We immediately notice that the performance of GLL1 and GLL2 is much better than suggested by Table 1. In particular, we observe that GLL2 is most often the best method in CPU time, in CG iterations, and, even more clearly, in function evaluations.

Finally, these results indicate very few reasons why one should use GLL1 instead of GLL2. The latter method is admittedly more complex to code, but is very often preferable to GLL1, as is indicated by the more detailed comparison between these methods presented in Table 3. In this table, we restrict our attention to 215 small and 94 large problems that were coherently solved by both methods, that is for which both methods converged to critical points with identical objective function values. Columns three to five of the tables indicate how many times each method was best and how many ties occurred. Columns six to ten describe the distribution of differences in values of the relevant criterion (iterations, function evaluations, CG iterations, or time) for the two methods, *when these values are different*: we report the minimum of this distribution (min), its first quartile (q1), its median (med), its third quartile (q3), and its maximum (max).

TABLE 3
Detailed comparison of GLL1 and GLL2.

	Performance	GLL1	tie	GLL2	GLL1-GLL2 (when different)				
	measure	wins		wins	min	q1	med	q3	max
215 small problems	Iterations	10	188	17	−775	−3	1	11	97
	Evaluations	21	27	167	−1561	2	3	4	294
	CG iterations	11	183	21	−3859	−3	12	93	1034
	Time	1	211	3	−1	3	3	4	4
94 large problems	Iterations	7	77	10	−16	−5	1	11	26
	Evaluations	12	14	67	−22	1	2	4	57
	CG iterations	6	76	12	−10676	−23	83	338	3262
	Time	4	79	11	−1131	−2	2	16	100

TABLE 4
Detailed comparison of BLS and GLL2.

	Performance	BLS	tie	GLL2	BLS-GLL2 (when different)				
	measure	wins		wins	min	q1	med	q3	max
206 small problems	Iterations	42	132	32	−694	−10	−2	9	499
	Evaluations	32	146	28	−6747	1	3	6	1696
	CG iterations	37	46	123	−53770	−23	4	51	18779
	Time	7	189	10	−151	−1	1	6	597
92 large problems	Iterations	22	59	11	−2633	−23	−7	7	192
	Evaluations	21	11	60	−9744	−1	2	6	514
	CG iterations	13	57	22	−649584	−96	43	3749	101631
	Time	10	64	18	−5250	−3	14	109	11066

Table 3 generally favors GLL2: this method indeed wins more often for all criteria and the medians of all difference distributions are positive. However, we observe that these distributions are rather dispersed. In particular, we note the striking difference of 10676 CG iterations in Table 3 for a large problem (MSQRTBLS (n = 1024)) in favor of GLL1, which results in an advantage of 1131 seconds for GLL1 on the same problem.

We conclude this analysis of the ideas of Grippo, Lampariello, and Lucidi presenting a detailed comparison of the best nonmonotone method, GLL2, with our basic linesearch algorithm (BLS). This comparison is given in Table 4 for the 206 small and 88 large problems coherently solved by both methods. The format of this table is identical to that of Table 3.

A few conclusions can be drawn from this comparison.

• The conclusions of Grippo, Lampariello, and Lucidi are confirmed in that a nonmonotone procedure appears to be often better than the basic linesearch method, and sometimes by a quite substantial amount. However, GLL1 and GLL2 occasionally perform quite badly compared with BLS. In particular, this is the case for problem FMINSURF (n = 1024), which corresponds to the large differences at the minimum of the distributions of BLS-GLL2 in Table 4.

• The dispersed character of most reported performance difference distributions indicates that a large part of the variations between methods, as appearing in Table 1, are explained by their behavior on a relatively small number of more difficult problems. The sometimes dramatic differences between BLS, GLL1, and GLL2 are caused by the strongly nonlinear nature of these problems. However, the trend is rather clear overall, indicating that one might indeed lose in performance by using a nonmonotone method on a very nonlinear problem, but with the potential of winning substantially more.

• It also results from Tables 3 and 4 that performance differences are relatively small for at least half the test problems used, but that these small differences also often favor GLL2 over both GLL1 and BLS.

- At variance with previously reported numerical experience, the results presented here dıstinguish the relative merits of nonmonotone techniques for small and large problems separately. As expected, the potential benefits and losses in CPU time are more important for the large problems, at least on our test set where the cost of computing the objective function and its derivatives is often small or modest compared with that of the involved linear algebra.

Furthermore, a remarkable effect seems to occur when combining the nonmonotone GLL1 and GLL2 methods and the truncated modified CG technique on some large problems. Because of the nonmonotonicity, the iterates do not appear to follow the bottom of curved valleys so closely for these two algorithms, but rather "stay higher on the slope." This implies that the average size of g_k is larger for these methods than for BLS, which in turn implies, because of (20), that the modified CG can be stopped sooner, which finally results in a smaller number of CG steps on average. As a large proportion of the computational effort for large problems is concentrated in this part of the algorithms, this effect yields a noticeable reduction in the overall CPU time for such problems.

3. A new nonmonotone variant.

3.1. Motivation and specification. In order to alleviate the difficulties of GLL2 revealed, for instance, by the FMINSURF problem, a more careful analysis of the performance of GLL2 was carried out for this case and a few others. It then appeared that the quadratic model implicit in Newton's method is often not well suited to represent the objective function, and that the stepsizes determined by the linesearch procedure are very often much smaller than one. Moreover, this behavior is accentuated for GLL1 and GLL2 compared with BLS. It is therefore natural to think of controlling the amount of nonmonotonicity allowed by a measure indicating how well adapted the truncated Newton step is to the true objective function. It also appeared in the tests that it is often advantageous to enforce monotonicity in the first few iterations. The purpose of this section is to show that these simple ideas can indeed produce additional benefits.

We therefore modified Algorithm GLL2 as follows.

- The equality $r_k = f(x_k)$ is enforced for the first 10 iterates.
- For all subsequent iterations, r_k is defined by

$$r_k = f(x_k) + \left(\prod_{j=1}^{j_k} \alpha_{k(\ell-j)}\right)^{\frac{1}{j_k}} [f_{\ell-j_k} - f(x_k)] \tag{25}$$

where

$$j_k = \arg \max_{j=1,\dots,p} f_{\ell-j} \tag{26}$$

and $\alpha_{k(\ell-j)}$ is the stepsize corresponding to the $(\ell - j)$th accepted step.

In this proposal, we have chosen to measure the adequacy of the Newton step by examining how close the past stepsizes α_k are to unity. The amount of nonmonotonicity allowed in the method is then controlled by this measure: it is indeed easy to see that r_k as given by (25) is equal to the value calculated by Algorithm GLL2 whenever the last j_k accepted stepsizes are equal to one (the coefficient of the second term in the right-hand side of (25) is their geometric mean). Because most stepsizes do not exceed one, it is typically smaller in other cases. In our implementation, p is chosen to be 10, as for GLL1 and GLL2. Of course, the convergence theory developed for GLL2 still holds for this variant, which we call the NMLS (nonmonotone linesearch) Algorithm.

TABLE 5
Cumulative statistics for BLS, GLL1, GLL2, and NMLS.

Method	206 small problems				92 large problems			
	iters	evals	CG-its	time	iters	evals	CG-its	time
BLS	8173	17926	134099	1752	5041	9231	541429	35233
GLL1	8249	22793	153943	1288	7564	17135	966247	22661
GLL2	8089	21637	151501	1300	7504	16795	968919	23586
NMLS	6620	13115	96571	1130	5057	11771	370379	20577

TABLE 6
Rankings for BLS, GLL1, GLL2, and NMLS on 206 small and 92 large problems.

Performance measure	Method	Small problems				Large problems			
		1st	2nd	3rd	4th	1st	2nd	3rd	4th
Iterations	BLS	169	17	17	7	76	7	6	3
	GLL1	148	34	24	0	65	9	15	3
	GLL2	154	35	13	4	65	14	10	3
	NMLS	168	31	7	0	70	17	4	1
Evaluations	BLS	44	118	24	20	26	48	9	9
	GLL1	39	124	25	18	21	53	10	8
	GLL2	155	29	19	3	63	20	6	3
	NMLS	166	28	8	4	67	16	8	1
CG iterations	BLS	154	25	15	12	67	9	10	6
	GLL1	150	34	20	2	69	9	12	2
	GLL2	159	31	13	3	71	15	4	2
	NMLS	161	31	14	0	72	13	6	1
Time	BLS	195	9	1	1	69	11	9	3
	GLL1	197	5	4	0	70	13	8	1
	GLL2	199	4	3	0	75	14	2	1
	NMLS	200	6	0	0	76	11	4	1

3.2. Numerical comparison. In order to assess this new variant, we also ran it on the set of test problems described above. This algorithm successfully solved 222 of the 226 small problems and 97 of the 101 large ones, a slightly better result than GLL2. We also computed values of the cumulative performance statistics for the 206 small and 92 large problems coherently solved by BLS, GLL1, GLL2, and NMLS. The comparison shown in Table 5 indicates a clear improvement on average for all criteria, compared with GLL1 and GLL2. In particular, the reduction in CG iterations is noticeable.

Examining the rankings presented in Table 6 confirms that NMLS is better than BLS, GLL1, or GLL2 not only on average, but also in a number of instances. This conclusion applies for both small and large problems. One interesting observation is that NMLS seems to require fewer iterations and fewer function evaluations than GLL2 on a significant number of problems.

If we now consider the detailed comparisons between NMLS, BLS, and GLL2 presented in Tables 7 and 8, the following additional comments can be made.

• Observations about the relative importance of more difficult problems in the average performance apply to NMLS as well as to the previously analyzed methods. However, one notices that the cases that are most unfavorable to NMLS compared with BLS (see Table 8) are much less so than those reported for GLL2 (see Table 4). We note, in particular, the very remarkable (although extreme) gain of 599569 CG iterations between GLL2 and NMLS on problem FMINSURF ($n = 1024$). This is not so surprising since NMLS is in fact designed

TABLE 7
Detailed comparison of GLL2 and NMLS.

	Performance	GLL2	tie	NMLS	GLL2-NMLS (when different)				
	measure	wins		wins	min	q1	med	q3	max
210 small problems	Iterations	15	158	37	−64	−1	4	13	475
	Evaluations	19	156	35	−189	−6	2	19	4390
	CG iterations	26	152	32	−577	−25	6	86	43688
	Time	4	200	6	−3	−3	1	2	104
95 large problems	Iterations	9	67	19	−29	−4	8	22	2208
	Evaluations	11	68	16	−73	−14	1	37	4738
	CG iterations	13	67	15	−8880	−285	9	281	599569
	Time	10	73	12	−941	−39	1	21	4810

TABLE 8
Detailed comparison of BLS and NMLS.

	Performance	BLS	tie	NMLS	BLS-NMLS (when different)				
	measure	wins		wins	min	q1	med	q3	max
213 small problems	Iterations	31	143	39	−243	−4	2	17	499
	Evaluations	24	27	162	−2357	2	3	6	1508
	CG iterations	27	137	49	−10082	−9	12	120	19664
	Time	3	199	11	−46	−3	2	8	635
93 large problems	Iterations	17	61	15	−425	−13	−1	13	190
	Evaluations	18	11	64	−5006	1	2	8	451
	CG iterations	12	60	21	−50015	−70	27	2602	92751
	Time	12	63	18	−440	−11	1	55	10124

to overcome the deficiency of GLL2 on that problem and similar ones. This big advantage in CG iterations results in a corresponding gain in CPU time.

- The smaller differences in performance (on the easier problems) also appear to favor NMLS over GLL2 and, more clearly, over BLS.

- As expected, the potential gain in CPU time between NMLS and GLL2 is more considerable for large problems and parallels, in this case, the potential gain in CG iterations. The same comment applies, although to a lesser extent, to the comparison between NMLS and BLS.

- We note that it is advantageous to use NMLS rather than BLS or even GLL2 when the objective function is costly to evaluate.

4. Preconditioning. We now examine the impact of preconditioning the iterative solution of the system (18) on the relative performance of BLS, GLL2, and NMLS. The preconditioner used is constructed by extracting from H_k its main 11-banded symmetric submatrix, which is then modified, if necessary, to ensure its positive definiteness. This modification is carried out according to the proposal of Schnabel and Eskow [21]. We note that this preconditioner has not been formally analyzed and, thus, that its effect is essentially unknown, although often beneficial. It is used in the default variant of trust-region-based LANCELOT package, although it is applied here in a context different from that of trust-region methods. This difference is quite significant because the typically longer steps determined by the modified CG procedure make the preconditioner's adequacy more crucial. The same preconditioner is also used by Conn et al. [8] in their study of iterated subspace minimization (ISM) methods. The preconditioned methods corresponding to BLS, GLL2, and NMLS are called PBLS, PGGL2, and PNMLS, respectively.

TABLE 9
Cumulative statistics for BLS, GLL2, NMLS, PBLS, PGLL2, and PNMLS.

Method	188 small problems				75 large problems			
	iters	evals	CG-its	time	iters	evals	CG-its	time
BLS	7649	16943	127062	1743	4682	8377	497420	32588
GLL2	6808	13878	91499	1246	4431	6139	281399	16315
NMLS	6057	10210	82298	1114	4228	5902	273230	17176
PBLS	11829	20569	85394	1868	6631	12832	304067	24419
PGLL2	10337	20689	65540	1193	5022	8117	217835	18786
PNMLS	7270	12931	61130	1351	5027	8234	241810	17689

TABLE 10
Detailed comparison of PBLS and PGLL2.

	Performance measure	PBLS wins	tie	PGLL2 wins	PBLS-PGLL2 (when different)				
					min	q1	med	q3	max
210 small problems	Iterations	51	111	48	−1694	−14	−1	13	1419
	Evaluations	43	140	27	−3347	1	3	8	2073
	CG iterations	43	106	61	−4627	−18	3	43	26194
	Time	14	184	12	−84	−1	−1	3	736
83 large problems	Iterations	10	51	22	−151	−3	7	56	3121
	Evaluations	9	60	14	−295	1	2	22	6334
	CG iterations	11	50	22	−45182	−24	67	533	79726
	Time	10	51	22	−5519	−1	6	19	7417

A first observation is that preconditioning slightly deteriorates reliability on our set of test problems: PBLS succeeds on 215 of the 226 small problems and both PGLL2 and PNMLS on 214. Of the 101 large problems, PBLS solves 89, PGLL2 solves 87, and PNMLS 86. In addition to the reasons for failure discussed above for the unpreconditioned case, we note that the search direction computed with the preconditioner sometimes causes the iterates to leave the region where the problem's functions are safely defined, causing arithmetic errors. Another effect is that the test (20) is not always adapted to the preconditioned system and may cause very early termination of the modified CG method, yielding a search direction identical or close to steepest descent. When this happens, convergence is typically very slow.

On the other hand, the preconditioner fulfills its purpose on most problems: it reduces the number of CG iterations requested for solution (although it might indeed deteriorate speed due to their increased computational cost). We illustrate this effect by comparing, in Table 9, the cumulative performances of the preconditioned algorithms and their unpreconditioned counterparts on the 188 small and 75 large problems that were coherently solved by all six methods. PNMLS appears to be the best of the two preconditioned methods on average for small problems, while PGLL2 takes the lead for the large ones, except, marginally, on CPU time.

The detailed comparisons between PBLS, PGLL2, and PNMLS are presented in Tables 10, 11, and 12.

These comparisons reinforce our previous conclusions on the relative merits of monotone versus nonmonotone methods. We note that the gap between preconditioned methods is smaller than that between unpreconditioned ones. We also see (Table 12) that PGLL2 seems to outperform PNMLS slightly. This may indicate that the additional caution introduced in NMLS may be less relevant when the computed search direction is closer to the true Newton step.

5. Conclusion and perspectives. In this paper, we have revisited Grippo, Lampariello, and Lucidi's proposals to use nonmonotone techniques in the framework of linesearch algorithms for unconstrained optimization. At variance with the original experiences reported

TABLE 11
Detailed comparison of PBLS and PNMLS.

	Performance	PBLS	tie	PNMLS	PBLS-PNMLS (when different)				
	measure	wins		wins	min	q1	med	q3	max
212 small problems	Iterations	35	131	46	−431	−12	2	12	1419
	Evaluations	32	28	152	−4513	1	3	7	2594
	CG iterations	32	127	53	−31999	−28	3	62	15700
	Time	9	191	12	−34	−2	0	2	505
84 large problems	Iterations	13	57	14	−285	−33	1	97	3121
	Evaluations	10	59	15	−3402	1	2	5	6334
	CG iterations	7	56	21	−27248	−2	109	439	64647
	Time	7	59	18	−3329	−2	5	97	7406

TABLE 12
Detailed comparison of PGLL2 and PNMLS.

	Performance	PGLL2	tie	PNMLS	PGLL2-PNMLS (when different)				
	measure	wins		wins	min	q1	med	q3	max
210 small problems	Iterations	37	133	40	−605	−6	1	12	2958
	Evaluations	45	129	36	−1571	−18	−2	16	5941
	CG iterations	45	127	38	−5738	−44	−2	34	10402
	Time	10	189	11	−231	−3	0	1	71
82 large problems	Iterations	14	57	11	−154	−34	−3	20	193
	Evaluations	16	56	10	−337	−69	−3	45	393
	CG iterations	14	57	11	−73112	−373	−3	91	32736
	Time	13	59	10	−2680	−10	−1	14	2189

by these authors, we used a (preconditioned) truncated CG technique to compute the search directions and applied the resulting methods to a large collection of both small- and large-dimensional test problems, totaling more than 2500 test runs.

Using the results of these experiments, we have assessed the potential of NMLS techniques as proposed by Grippo, Lampariello, and Lucidi. The effects of problem size and preconditioning have also been specifically pointed out. We finally proposed a new variant that uses similar techniques.

The general conclusions of these tests and proposals can be summarized as follows.

- The impact of NMLS in the context of truncated CG methods is clear, and most noticeable on the more difficult and nonlinear problems.
- This impact is often favorable, in that nonmonotone methods show superior efficiency when compared with standard linesearch algorithms. This is especially true if preconditioning is not used. However, unfavorable problem-dependent effects also appear, but seem to be outweighed by the favorable ones.
- As expected, the differences in CPU time between monotone and nonmonotone methods are most important for the large problems. This distinction, however, is irrelevant when the number of problem function evaluations is considered: the gains provided by some of the analyzed nonmonotone methods are consistent, regardless of problem size.
- In the absence of preconditioning, the new proposed variant appears to yield the highest potential for substantial efficiency gains while best limiting the probability of losses, at least on the set of test problems used in this comparison. However, this advantage seems to disappear when preconditioning is applied.

As is the case for all practical comparisons between optimization algorithms, the results presented here do not pretend to be either complete or decisive. Further confirmation by continued experience is of course necessary for a true validation of NMLS techniques, but the trends outlined in this study are definitely encouraging. The application of these techniques in the context of merit function minimization for the solution of constrained problems and in the framework of ISM [8] are of particular interest. Finally, the extension of the nonmonotone approach to a trust-region algorithm is the object of ongoing research.

Acknowledgments. The author is indebted to N. Gould for kindly providing a Fortran 77 code for the modified CG technique and to S. Lucidi, M. Roma, and A. Sartenaer for many useful comments. Thanks are also due to a referee who suggested pursuing the tests with PGLL2, which resulted in a better appraisal of the methods compared in this paper.

REFERENCES

[1] M. Arioli, T. F. Chan, I. S. Duff, N. I. M. Gould, and J. K. Reid, *Computing a Search Direction for Large-Scale Linearly Constrained Nonlinear Optimization Calculations*, Technical Report TR/PA/93/34, CERFACS, Toulouse, France, 1993.

[2] I. Bongartz, A. R. Conn, N. Gould, and Ph. L. Toint, CUTE: *Constrained and unconstrained testing environment*, ACM Trans. Math. Software, 21 (1995), pp. 123-160.

[3] J. V. Burke, J. J. Moré, and G. Toraldo, *Convergence properties of trust region methods for linear and convex constraints*, Math. Programming Ser. A, 47 (1990), pp. 305-336.

[4] R. H. Byrd, D. C Liu, and J. Nocedal, *On the behaviour of broyden's class of quasi-Newton methods*, SIAM J. Control Optim., 2 (1992), pp. 533-557.

[5] R. M. Chamberlain, C. Lemaréchal, H. C. Pedersen, and M. J. D. Powell, *The watchdog technique for forcing convergence in algorithms for constrained optimization*, Math. Programming Studies, 16 (1982), pp. 1-17.

[6] A. R. Conn, N. I. M. Gould, and Ph. L. Toint, *Global convergence of a class of trust region algorithms for optimization with simple bounds*, SIAM J. Numer. Anal., 25 (1988), pp. 433-460; 26 (1989), pp. 764-767.

[7] ———, LANCELOT*: A Fortran package for large-scale nonlinear optimization (Release A)*, Springer Series in Computational Mathematics 17, Springer-Verlag, Heidelberg, Berlin, New York, 1992.

[8] A. R. Conn, N. Gould, A. Sartenaer, and Ph. L. Toint, *On Iterated-Subspace Minimization Methods for Nonlinear Optimization*, Technical Report 94/13, Department of Mathematics, Facultes Universitaires Notre Dame de la Paix, Namur, Belgium, 1994.

[9] J. E. Dennis and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1983.

[10] P. E. Gill and W. Murray, *Newton-type methods for unconstrained and linearly constrained optimization*, Math. Programming, 28 (1974), pp. 311-350.

[11] P. E. Gill, W. Murray, D. B. Ponceléon, and M. A. Saunders, *Preconditioners for indefinite systems arising in optimization*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 292-311.

[12] A. Griewank and Ph. L. Toint, *Local convergence analysis for partitioned quasi-Newton updates*, Numer. Math., 39 (1982), pp. 429-448.

[13] L. Grippo, F. Lampariello, and S. Lucidi, *A nonmonotone line search technique for Newton's method*, SIAM J. Numer. Anal., 23 (1986), pp. 707-716.

[14] ———, *A truncated Newton method with nonmonotone line search for unconstrained optimization*, J. Optim. Theory Appl., 60 (1989), pp. 401-419.

[15] ———, *A class of nonmonotone stabilization methods in unconstrained optimization*, Numer. Math., 59 (1991), pp. 779-805.

[16] Ch. Kanzow, *Some Tools Allowing Interior-Point Methods To Become Noninterior*, Technical Report A-79, Institute of Applied Mathematics, University of Hamburg, D-20146 Hamburg, Germany, 1994.

[17] J. E. Koontz, R. B. Schnabel, and B. E. Weiss, *A modular system of algorithms for unconstrained minimization*, ACM Trans. Math. Software, 11 (1985), pp. 419-440.

[18] D. C. Liu and J. Nocedal, *On the limited memory BFGS method for large scale optimization*, Math. Programming Ser. B, 45 (1989), pp. 503-528.

[19] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.

[20] T. Schlick, *Modified Cholesky factorizations for sparse preconditioners*, SIAM J. Sci. Statist. Comput., 14 (1993), pp. 424-445.

[21] R. B. Schnabel and E. Eskow, *A new modified Cholesky factorization*, SIAM J. Sci. Statist. Comput., 11 (1991), pp. 1136-1158.

[22] R. B. Schnabel and T.-T. Chow, *Tensor methods for unconstrained optimization using second derivatives*, SIAM J. Control Optim., 1 (1991), pp. 293-315.

[23] Ph. L. Toint, VE08AD, *a Routine for Partially Separable Optimization with Bounded Variables*, Harwell Subroutine Library, 2, 1983.

[24] ———, VE10AD, *a Routine for Large Scale Nonlinear Least Squares*, Harwell Subroutine Library (Release 2), Harwell Laboratory, Harwell, UK, 1987.

[25] ———, *Global convergence of a class of trust region methods for nonconvex minimization in Hilbert space*, IMA J. Numer. Anal., 8 (1988), pp. 231-252.

[26] ———, *An Assessment of Non-Monotone Linesearch Techniques for Unconstrained Optimization: The Complete Numerical Results*, Technical Report 94/15, Department of Mathematics, Facultes Universitaires Notre Dame de la Paix, Namur, Belgium, 1994. This report can be obtained by anonymous `ftp` from the directory `pub/reports` on `thales.math.fundp.ac.be` (internet 138.48.4.14).

SIAM J. SCI. COMPUT.
Vol. 17, No. 3, pp. 740–749, May 1996

A REGULARIZATION PARAMETER IN DISCRETE ILL-POSED PROBLEMS*

TERESA REGIŃSKA†

Abstract. The Tikhonov regularization method for discrete ill-posed problems is considered. For the practical choice of the regularization parameter α, some authors use a plot of the norm of the regularized solution versus the norm of the residual vector for all α considered. This paper contains an analysis of the shape of this plot and gives a theoretical justification for choosing the regularization parameter so it is related to the "L-corner" of the plot considered in the logarithmic scale. Moreover, a new criterion for choosing α is introduced (independent of the shape of the plot) which gives a new interpretation of the "corner criterion" mentioned above. The existence of "L-corner" is discussed.

Key words. discrete ill-posed problems, least squares solution, Tikhonov regularization, regularization parameter

AMS subject classifications. 65F20, 65J20

1. Introduction. A linear integral equation of the first kind in $L^2(I)$ with a smooth kernel is a classical example of an ill-posed problem. A solution of this equation, if it exists, does not continuously depend on the right-hand side and may not be unique. A discretization of such a problem leads to a matrix equation in $\mathbb{C}^m$,

$$Ku = f, \tag{1.1}$$

where K is an $m \times n$ matrix with a large condition number, $m \geq n$. A linear least squares solution of the system (1.1) is a solution to the problem

$$\min_{u \in \mathbb{C}^n} \|Ku - f\|^2, \tag{1.2}$$

where the Euclidian vector norm in $\mathbb{C}^m$ is used. Using the notation introduced in other papers (e.g., [3, 8]), we say that the algebraic problems (1.1) and (1.2) are discrete ill-posed problems.

There are many papers concerning numerical methods for solving ill-posed problems in function spaces (cf. [5, 10, 16, 17, 12, 13]), as well as for solving discrete ill-posed problems (cf. [2, 3, 6, 8, 11]). These methods are based on so-called regularization methods. The most well known is the Tikhonov approach, which consists of replacing the least squares problem (1.2) by the problem of a suitably chosen Tikhonov functional. The simplest version of this method has the form

$$\min_{u \in \mathbb{C}^n} \{\|Ku - f\|^2 + \alpha^2 \|u\|^2\}, \tag{1.3}$$

where $\alpha \in R$ is called the regularization parameter.

An important and still current problem is a proper choice of the regularization parameter. There are several possible strategies that depend on additional information concerning the considered problem and its solution. For instance, in order to apply the well-known discrepancy principle (cf. [5, 10, 16, 17]), which is an a posteriori strategy for choosing α as a function of error level, the input error level must be known.

Another strategy based on a priori knowledge of a structure of input error, namely, when error in f can be considered to be uncorrelated zero mean random variables with a common variance (white noise), is the generalized cross-validation method [4, 18]. Surveys of the different methods can be found in [2, 7, 8, 18].

In [7] and [8] the authors investigated another practical method for choosing α in the case when data are noisy. The method is based on a plot of the norm of the regularized

*Received by the editors July 21, 1993; accepted for publication (in revised form) December 19, 1994.

†Institute of Mathematics, Polish Academy of Sciences, 00-950 Warsaw, POB 137, Poland (reginska@impan.gov.pl).

solution versus the norm of the corresponding residual. The idea of the proposed L-curve corner criterion is to choose the regularization parameter related to a characteristic L-shaped "corner" of the graph.

Unfortunately, the location of the "corner" is dependent on the scale in which the L-curve is considered, and in some scales it may not appear. Moreover, in the aforementioned papers, some statements concerning a characterization of the L-curve have only intuitive justification and are not true in general. Nevertheless, the idea of the method is intuitively clear and numerical results (cf. [7]) confirm the utility of it when the logarithmic scale is used.

The present paper is devoted to more rigorous analysis of the method mentioned above. Moreover, an interpretation of the L-curve corner criterion in terms of minimization is given.

In §2, we give a short review of properties of the norm of regularized solution (in the Tikhonov sense (1.3)), the residual norm, and the corresponding plot in different scales. In §3, we formulate another criterion for choosing α which is independent of the shape of L. Next, we prove that this criterion is equivalent to the L-curve corner criterion related to the plot in the logarithmic scale. In §4, the problem of existence of an "L-corner" indicating the regularization parameter α is discussed.

2. L-curve. It was observed in [9] and extended in [7, 8] that a simple graphic presentation of the curve

$$x(\alpha) = \|Ku_\alpha - f\|^2, \tag{2.1}$$

$$y(\alpha) = \|u_\alpha\|^2, \tag{2.2}$$

where u_α is the solution of (1.3), can be very useful in studying the least square problem connected with it. In this section we give a short review of the properties of this curve in different scales. Some of them were formulated in other papers (cf. [7, 8, 14]). According to the notation introduced in [7], by L-curve we will mean

$$\{x(\alpha), y(\alpha)\}_{\alpha\in(0,\infty)}. \tag{2.3}$$

The L-curve in ϕ-scale will be denoted by

$$L_\phi = \{\phi(x(\alpha)), \phi(y(\alpha))\}_{\alpha\in(0,\infty)}. \tag{2.4}$$

$\phi = \sqrt{\ }$ and $\phi = \ln$ will be considered.

The following lemma for $\phi = \sqrt{\ }$ describes a standard property of the Tikhonov regularization method (cf. [9, Thm. 25.99]).

LEMMA 1. *If $\phi : R^+ \to R$ is a positive, increasing function then any point (ξ, η) on the L_ϕ is a solution of the following two inequality-constrained least squares problems:*

$$\xi = \min \phi(\|Ku - f\|^2) \textit{ subject to } \phi(\|u\|^2) \le \eta,$$

$$\eta = \min \phi(\|u\|^2) \textit{ subject to } \phi(\|Ku - f\|^2) \le \xi.$$

For the further investigation of $x(\alpha)$ and $y(\alpha)$ we shall apply the singular value decomposition for the matrix K. This is a well-known approach for least squares problems (cf. [18]).

If $K \in \mathbb{C}^{m,n}$ is a matrix of rank r, then there exist unitary matrices $U \in \mathbb{C}^{m,m}$ and $V \in \mathbb{C}^{n,n}$ such that

$$K = U\Sigma V^*, \ \Sigma = \begin{pmatrix} \Sigma_r & 0 \\ 0 & 0 \end{pmatrix},$$

where $\Sigma \in R^{m,n}$, $\Sigma_r = \operatorname{diag}(\sigma_1, \dots, \sigma_r)$, and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$. The σ_i are called the singular values of K and the ith column vectors u_i, v_i of U and V, respectively, are the left and right singular vectors corresponding to σ_i, $i = 1, \dots, r$.

The least squares solution of (1.1) of the minimal norm is the minimal norm solution of the normal equation $K^*Ku = K^*f$, and thus if

$$f = \sum_{i=1}^{m} f_i u_i, \quad \text{where } f_i = u_i^* f, \; i = 1, \dots, m \tag{2.5}$$

then

$$u = \sum_{i=1}^{r} \frac{f_i}{\sigma_i} v_i. \tag{2.6}$$

For a discrete ill-posed problem (just the case considered here) the singular values σ_i tend with i very rapidly to zero. Due to the errors in the right-hand side ($f = \bar{f} + e$), for which it cannot be assumed that $u_i^* e$, $i = 1, \dots, r$, tend to zero faster than σ_i, the solution u is perturbed mainly by contributions corresponding to the small singular values.

The regularized solution u_α of (1.3) satisfies the normal equation $K^*Ku + \alpha^2 u = K^*f$, thus

$$u_\alpha = \sum_{i=1}^{r} \frac{\sigma_i f_i}{\sigma_i^2 + \alpha^2} v_i. \tag{2.7}$$

Because of $\alpha > 0$, the problem of computing u_α becomes less ill-conditioned than that of computing u, i.e., an influence of errors corresponding to the small singular values becomes smaller.

It can easily be found that

$$y(\alpha) = \|u_\alpha\|^2 = \sum_{i=1}^{r} \frac{\sigma_i^2 |f_i|^2}{(\sigma_i^2 + \alpha^2)^2}. \tag{2.8}$$

Moreover, denoting $f_\perp = \sum_{i=r+1}^{m} f_i u_i$, we get

$$Ku_\alpha - f = -\left(\sum_{i=1}^{r} \frac{\alpha^2 f_i}{\sigma_i^2 + \alpha^2} u_i + f_\perp \right).$$

Thus

$$x(\alpha) = \sum_{i=1}^{r} \frac{\alpha^4 |f_i|^2}{(\sigma_i^2 + \alpha^2)^2} + \|f_\perp\|^2. \tag{2.9}$$

From this it follows that for $f \neq f_\perp$

$$\frac{dx}{d\alpha} = 4\alpha^3 \sum_{i=1}^{r} \frac{|f_i|^2 \sigma_i^2}{(\sigma_i^2 + \alpha^2)^3} > 0, \tag{2.10}$$

which means that the function $x(\alpha)$ is strictly increasing. Thus there exists a function $\alpha(x)$ inverse to x, and the curve $\{x(\alpha), y(\alpha)\}_{\alpha \in R^+}$ has only one branch which can be represented as the function $y(x) = y(\alpha(x))$.

LEMMA 2. *If x, y are defined by* (2.1), (2.2) *then y as a function of x is decreasing and strictly convex.*

Proof. The function $x : (0, \infty) \to (\|f_\perp\|^2, \|f\|^2)$ is increasing; thus the inverse function

$$\alpha(x) : (\|f_\perp\|^2, \|f\|^2) \to (0, \infty)$$

is also increasing. From the formula (2.8)

$$\frac{dy}{d\alpha} = -4\alpha \sum_{i=1}^{r} \frac{\sigma_i^2 |f_i|^2}{(\sigma_i^2 + \alpha^2)^3}. \tag{2.11}$$

Taking into account (2.10) we get

$$\frac{dy}{dx} = -\frac{1}{\alpha^2}. \tag{2.12}$$

Thus $y(\cdot)$ is decreasing and $\frac{dy}{dx} \to -\infty$ when $x \to \|f_\perp\|^2$; $\frac{dy}{dx} \to 0$ when $x \to \|f\|^2$. We have

$$\frac{d^2y}{dx^2} = -\frac{d}{dx}\frac{1}{\alpha^2} = \frac{2}{\alpha^3}\frac{d\alpha(x)}{dx} > 0,$$

since $\alpha(x)$ is increasing. This means that $y(x)$ is strictly convex. □

From formulas (2.8) and (2.6) it follows that $y(\alpha) \to \|u\|^2$ when $\alpha \to 0$. Moreover, $y(\alpha) \to 0$ when $\alpha \to \infty$.

REMARK 1. *The L-curve remains decreasing in any differentiable strictly monotonic scale ϕ, among others, for $\phi = \ln$ and $\phi = \sqrt{}$. Using the notation $\eta = \phi(y(\alpha))$, $\xi = \phi(x(\alpha))$,*

$$\frac{d\eta}{d\xi} = -\frac{1}{\alpha^2}\frac{\phi'(y(\alpha))}{\phi'(x(\alpha))}. \tag{2.13}$$

REMARK 2. *A change of the scale ϕ does not preserve the convexity of the L-curve. For instance, the parametric curve $L = \{\alpha^4/(1+\alpha^2)^2, 1/(1+\alpha^2)^2\}_{\alpha\in(0,\infty}$ (which is a particular case of an L-curve defined by* (2.8), (2.9) *for $r = 1$, $\sigma_1 = 1$, $f_1 = 1$, $f_i = 0$ for $i \neq 1$) in the scale $\phi = \sqrt{}$ is the straight line interval with ends $\{1, 0\}$ and $\{0, 1\}$, while in the logarithmic scale it becomes a strictly concave curve.*

Because of the fact that the shape of the L-curve in the scales $\phi = \sqrt{}$ and $\phi = \ln$ determines the practical choice of a proper α (cf. [7], [8]), we present the following lemmas to give some information about that shape.

First, let us consider the L-curve in the logarithmic scale.

LEMMA 3. *If $f_\perp = 0$, the L-curve in the logarithmic scale is strictly concave for $\alpha \in (0, \sigma_r)$ and for $\alpha \in (\sigma_1, \infty)$. If $f_\perp \neq 0$ then this curve is convex for $\alpha \in (0, \epsilon)$ where ϵ is sufficiently small, and remains concave for $\alpha \in (\sigma_1, \infty)$.*

Proof. According to (2.13)

$$\frac{d\eta}{d\xi} = -\frac{1}{\alpha^2}\frac{x}{y}.$$

Since $x = e^\xi$, $y = e^\eta$, we easily find that

$$\frac{d^2\eta}{d\xi^2} = -\frac{x}{\alpha^4 y^2}\left[\alpha^2 y + x - 2\alpha y \frac{1}{\frac{d\xi}{d\alpha}}\right],$$

and by (2.10)

$$\frac{d\xi}{d\alpha} = \frac{4\alpha^3}{x}z, \quad \text{where } z = \sum_{i=1}^{r} \frac{|f_i|^2\sigma_i^2}{(\sigma_i^2+\alpha^2)^3}.$$

Thus

$$\frac{d^2\eta}{d\xi^2} = -\frac{x^2}{\alpha^6 y}\left(\frac{\alpha^4}{x} + \frac{\alpha^2}{y} - \frac{1}{2z}\right).$$

Let us observe that for $\alpha \leq \sigma_r$

$$\frac{2\sigma_i^2}{\sigma_i^2+\alpha^2} \geq 1, \quad i = 1, \dots, r,$$

and for $\alpha \geq \sigma_1$

$$\frac{2}{\sigma_i^2+\alpha^2} \geq \frac{1}{\alpha^2}, \quad i = 1, \dots, r.$$

It follows that for $\alpha \leq \sigma_r$ and $f_\perp = 0$ we have $2z \geq \frac{x}{\alpha^4}$, while for $\alpha \geq \sigma_1$ (independently of $f_\perp$) $2z \geq \frac{y}{\alpha^2}$. This implies that in both cases

$$\frac{\alpha^4}{x} + \frac{\alpha^2}{y} - \frac{1}{2z} > 0, \quad \text{i.e.,} \quad \frac{d^2\eta}{d\xi^2} < 0,$$

which ends the proof. □

From Remark 2 it follows that in the case $f_i = 0$, $\forall i \neq 1$, the curve $\mathrm{L}_{\ln}$ is strictly concave. Due to continuity this curve remains concave for sufficiently small perturbation of coefficients. The following lemma gives a sufficient condition (in the case of two components of f) under which the $\mathrm{L}_{\ln}$-curve does not have a "corner property."

LEMMA 4. *Let $i_1 < i_2$ be such that $f_i \neq 0$ for $i \in \{i_1, i_2\}$ and $f_i = 0$ for $i \notin \{i_1, i_2\}$. If*

$$\frac{|f_{i_2}|}{\sigma_{i_2}^2} < \frac{|f_{i_1}|}{\sigma_{i_1}^2}$$

then the $L_{\ln}$-curve is strictly concave.

Proof. The proof follows from simple calculations concerning the second derivative $\frac{d^2\xi}{d\eta^2}$, where $\xi = \ln x(\alpha)$ and $\eta = \ln y(\alpha)$. The sign of $\frac{d^2\xi}{d\eta^2}$ is equal to the sign of a certain polynomial of degree 4 with respect to α^2. Under the assumption of Lemma 4, all coefficients of this polynomial are positive. □

A relation between the magnitude of quotients $\frac{|f_i|}{\sigma_i}$ and the shape of the corresponding $L_{\log}$-curve is illustrated in Figs. 1 and 2 for the case of two components of f. Figure 1 presents $L_{\log}$-curves for $\sigma_1 = 1$, $\sigma_2 = 0.01$, $f_1 = 1$, and various values of f_2: 0.001, 0.03, 0.1. Figure 2 shows the changing shape of the $L_{\log}$-curve when $\|f_\perp\|$ grows. For the case $\sigma_1 = 1$, $\sigma_2 = 0.01$, $f_1 = 1$, $f_2 = 0.05$ the values $\|f_\perp\|^2 = 0$, 0.001, 0.01 are taken. The "+" sign denotes the points of the $L_{\log}$-curves corresponding to $\alpha = \sigma_2$.

Finally, let us consider a shape of an L-curve in the square-root scale. In the particular case when the decomposition of the right-hand side in the basis $\{u_i\}$ has only two nonzero components we have the following result.

LEMMA 5. *Let $i_1 < i_2$ be such that $f_i \neq 0$ for $i \in \{i_1, i_2\}$ and $f_i = 0$ for $i \notin \{i_1, i_2\}$. Then the parametric curve*

$$L_{\sqrt{}} = \{\|Ku_\alpha - f\|, \|u_\alpha\|\}_{\alpha\in(0,\infty)}$$

is convex.

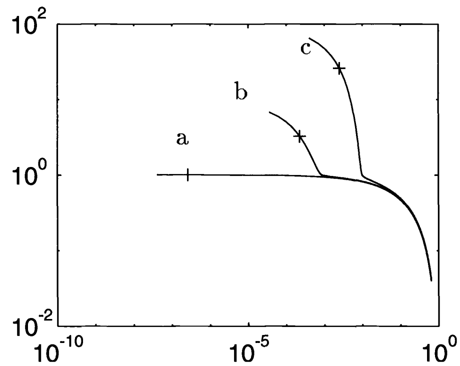


FIG 1. $L_{\log}$*-curves for* $f_\perp = 0$, $\sigma_1 = 1, \sigma_2 = 0.01$, *and*
(a) $\frac{f_1}{\sigma_1} = 1$, $\frac{f_2}{\sigma_2} = 0.1$;
(b) $\frac{f_1}{\sigma_1} = 1$, $\frac{f_2}{\sigma_2} = 3$;
(c) $\frac{f_1}{\sigma_1} = 1$, $\frac{f_2}{\sigma_2} = 10$.

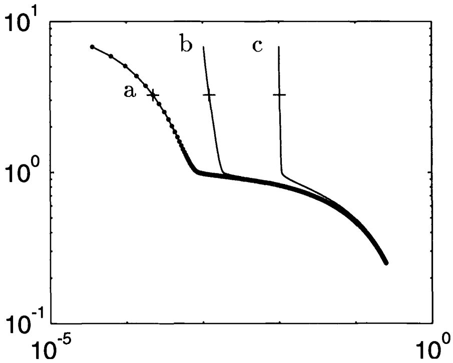


FIG 2. $L_{\log}$*-curves for different* $f_\perp$, $\sigma_1 = 1, \sigma_2 = 0.01, f_1 = 1, f_2 = 0.05$,
(a) $\|f_\perp\|^2 = 0$;
(b) $\|f_\perp\|^2 = 0.01$;
(c) $\|f_\perp\|^2 = 0.001$.

Proof. Let $\{\xi, \eta\} \in L_{\sqrt{}}$, i.e., $\xi = \sqrt{x(\alpha)}$, $\eta = \sqrt{y(\alpha)}$. According to (2.13), $\frac{d\eta}{d\xi} = -\frac{1}{\alpha^2}\sqrt{\frac{x}{y}}$. If $\sigma_{i_1} = \sigma_{i_2}$ then $\frac{d\eta}{d\xi} = -\frac{1}{\sigma_{i_1}} = const.$

Let $\sigma_{i_1} > \sigma_{i_2}$. Using the formulas (2.8) and (2.9) we get

$$\frac{d\eta}{d\xi} = -\sqrt{\frac{|f_{i_1}|^2(\sigma_{i_2}^2+\alpha^2)^2 + |f_{i_2}|^2(\sigma_{i_1}^2+\alpha^2)^2}{|f_{i_1}|^2\sigma_{i_1}^2(\sigma_{i_2}^2+\alpha^2)^2 + |f_{i_2}|^2\sigma_{i_2}^2(\sigma_{i_1}^2+\alpha^2)^2}} \tag{2.14}$$

and consequently,

$$f_{i_1}{}^2\left(\sigma_{i_2}^2+\alpha^2\right)^2\left(1-\sigma_{i_1}^2\left(\frac{d\eta}{d\xi}\right)^2\right) = |f_{i_2}|^2\left(\sigma_{i_1}^2-\alpha^2\right)^2\left(\sigma_{i_2}^2\left(\frac{d\eta}{d\xi}\right)^2-1\right). \tag{2.15}$$

Suppose that $\frac{d\eta}{d\xi}(\alpha_1) = \frac{d\eta}{d\xi}(\alpha_2)$. From (2.15)

$$\frac{f_{i_1}{}^2\left(\sigma_{i_2}^2+\alpha_1^2\right)^2}{|f_{i_2}|^2\left(\sigma_{i_1}^2-\alpha_1^2\right)^2} = \frac{f_{i_1}{}^2\left(\sigma_{i_2}^2+\alpha_2^2\right)^2}{|f_{i_2}|^2\left(\sigma_{i_1}^2-\alpha_2^2\right)^2}.$$

Thus

$$\sigma_{i_1}^2\alpha_1^2 + \sigma_{i_2}^2\alpha_2^2 = \sigma_{i_1}^2\alpha_2^2 + \sigma_{i_2}^2\alpha_1^2,$$

which is true only if $\alpha_1 = \alpha_2$.

The conclusion is that $\frac{d\eta}{d\xi}$ is a continuous one-to-one function of α, i.e., it is strictly monotonic. If $\frac{d\eta}{d\xi}$ is monotonically decreasing, the function $\eta(\xi)$ is strictly convex. On the other hand, if $\frac{d\eta}{d\xi}$ is monotonically increasing then $\eta(\xi)$ is concave.

Let us introduce an auxiliary function $f : R^+ \times R^+ \to R^+$,

$$f(t, s) := \frac{t+s}{t\sigma_{i_1}^2 + s\sigma_{i_2}^2}.$$

If we put $t_o = |f_{i_1}|^2 \frac{\sigma_{i_2}^4}{\sigma_{i_1}^4}$, $s_o = |f_{i_2}|^2$, $t_\infty = |f_{i_1}|^2$, then from (2.14)

$$\lim_{\alpha\to 0} \frac{d\eta}{d\xi} = -\sqrt{f(t_o, s_o)}; \quad \lim_{\alpha\to\infty} \frac{d\eta}{d\xi} = -\sqrt{f(t_\infty, s_o)}.$$

Since for $\sigma_{i_1} > \sigma_{i_2}$ we have $\frac{\partial f}{\partial t} < 0$, the function f is decreasing in t. This and $t_0 < t_\infty$ imply $f(t_o, s_o) > f(t_\infty, s_o)$. Thus

$$\lim_{\alpha\to 0} \frac{d\eta}{d\xi} < \lim_{\alpha\to\infty} \frac{d\eta}{d\xi},$$

which means that $\frac{d\eta}{d\xi}$ is increasing, i.e., η is strictly convex. □

Moreover, it can be proved (cf. [14, Thm. 1]) that in the case of arbitrary components f_i of f the maximum of curvature of the L-curve in square-root scale is always attained inside the interval $[\frac{1}{\sigma 1}, \infty)$. The above results suggest that the maximum of curvature of $L_{\sqrt{}}$ may not indicate a proper regularization parameter α.

3. A local minimum criterion. Let us consider a discrete ill-posed problem (1.2) and its solution u (2.6) and a regularized solution u_α (2.7) by means of Tikhonov regularization (1.3).

The right-hand side f in (1.1) is assumed to be contaminated with measurement errors. While the unperturbed right-hand side $\bar{f}$ satisfies the discrete Picard condition (i.e., the Fourier coefficients $u_i^* \bar{f}$ tend to zero faster than σ_i), the perturbation vector $f - \bar{f}$ does not need to satisfy such a condition. A consequence is a large influence of the errors $f_i - \bar{f}_i$ corresponding to small singular values on the solution norm (cf. (2.6)). Any $\alpha > 0$ reduces the norm of u. The problem is how to choose a proper α.

1. The first postulate is to choose α such that $y(\alpha)$ (cf. (2.8)) is not too large, say $y(\alpha) \le y(\alpha_o)$, i.e., $\alpha \ge \alpha_o$.

2. On the other hand, another natural postulate is to have a small norm of residuum, i.e., $x(\alpha) \le x(\alpha_1)$, which means that $\alpha \le \alpha_1$.

Because of the fact that $x(\alpha)$ is increasing while $y(\alpha)$ is a decreasing function of α, it is reasonable to consider the function $\Psi(\alpha) = x(\alpha)y(\alpha)$, or more generally,

$$\Psi_\lambda(\alpha) = x(\alpha)y^\lambda(\alpha), \quad \text{where } \lambda > 0, \tag{3.1}$$

and look for its minima over the interval $[\alpha_o, \alpha_1]$. Since $y(\alpha) \to 0$ as $\alpha \to \infty$, and if $f_\perp = 0$ then $x(0) = 0$, the global infimum is always equal to zero. It is attained at ∞ and, if $f_\perp = 0$, at 0. The question is about existence of local minima points inside the open interval $(0, \infty)$ and about their characterization.

The following theorem shows that the local minimum of Ψ_λ has a simple geometric interpretation, very similar to the so-called L-curve corner described and applied in [7] and [8], where the L-corner means the point of the L-curve with the maximum curvature.

THEOREM 1. *The function Ψ_λ has a local minimum at $\bar{\alpha}$ if and only if*

(i) *the L_{ln}-curve is tangent at the point $\{\xi(\bar{\alpha})\eta(\bar{\alpha})\}$ to a straight line $\eta + \lambda\xi = const$, and*

(ii) *the L_{ln} is convex in a neighbourhood of $\{\xi(\bar{\alpha})\eta(\bar{\alpha})\}$.*

Proof. Using the notation $\xi(\alpha) = \ln x(\alpha)$, $\eta(\alpha) = \ln y(\alpha)$, we have

$$\Psi(\alpha) = \exp(\xi(\alpha) + \lambda\eta(\alpha)).$$

Since $\frac{d\xi}{d\alpha} > 0$, the necessary condition for a local extreme of Ψ is

$$\frac{d\eta}{d\xi}(\alpha) + \frac{1}{\lambda} = 0, \tag{3.2}$$

which is equivalent to the condition of a tangent of L_{ln} to the straight line $\xi + \lambda\eta = const$.

If $\Psi(\alpha)$ attains a local minimum at $\hat{\alpha}$ then $\xi(\alpha) + \lambda\eta(\alpha)$ also attains a local minimum at $\hat{\alpha}$. It means that for a certain neighbourhood of $\hat{\alpha}$ all points $\{\xi(\alpha), \eta(\alpha)\}$ of the $L_{\ln}$-curve lay above the straight line $\xi + \lambda\eta = \xi(\hat{\alpha}) + \lambda\eta(\hat{\alpha})$. Thus condition (ii) is satisfied. The inverse implication is evident. □

REMARK 3. *When $\hat{\alpha} > 0$ is determined by the maximum of curvature of $L_{\ln}$ then there exists $\lambda > 0$ such that*

$$\hat{\alpha} = \arg loc \min_{0<\alpha<\infty} \Psi_\lambda(\alpha).$$

Inversely, if $\Psi_\lambda(\alpha)$ attains at $\hat{\alpha}$ a local minimum then the maximum curvature criterion indicates $\hat{\alpha} > 0$.

REMARK 4. *Let ϕ be an arbitrary one from the considered scales (I, $\surd$, ln). If $\hat{\alpha}$ is determined by conditions* (i) *with $\lambda = 1$, and* (ii) *(Theorem* 1*) applied to L_ϕ then*

$$\frac{d\eta}{d\xi}(\hat{\alpha}) = \frac{d\xi}{d\eta}(\hat{\alpha}), \text{ where } \eta(\alpha) = \phi(y(\alpha)),\ \xi(\alpha) = \phi(x(\alpha)).$$

For $\phi = I$, these conditions give exactly one $\hat{\alpha} = 1$ (from (2.12)*).*

In the general case the problem of uniqueness of a local minimum of Ψ_λ inside $(0, \infty)$ is open. The same question applies to the L-corner.

4. When do the L-curve criteria indicate $\alpha > 0$? By L-curve criteria we mean (i) Hansen's criterion [7, 8] taking α as the point at which the L-curve has the maximum curvature, and (ii) a choice of α according to conditions (i) and (ii) of Theorem 1 (the local minimum criterion).

The case described in Lemma 4 shows that the $L_{\ln}$-curve may have no L-corners, i.e., local minima of Ψ_λ may not exist, as well as that the maximum of curvature of $L_{\ln}$ may be attained at $\alpha = 0$. We are going to explain when the local minimum of Ψ exists (i.e., when L-corner of $L_{\ln}$ appears) and what it means when there are no local minima.

First, let us formulate the following auxiliary lemma.

LEMMA 6. *There exists a local minimum of $\Psi(\alpha)$ at $\hat{\alpha} \in (0, \infty)$ iff in the case $f_\perp = 0$*

$$(4.1) \qquad \exists\alpha_1, \alpha_2 \quad 0 \le \alpha_1 < a_2, \text{ such that } x(\alpha_1) \ge \alpha_1^2 y(\alpha_1),\ x(\alpha_2) \le \alpha_2^2 y(\alpha_2),$$

or in the case $f_\perp \neq 0$

$$(4.2) \qquad \exists\alpha_2 > 0 \text{ such that } x(\alpha_2) \le \alpha_2^2 y(\alpha_2),$$

Proof. Let $\tilde{\Psi}(x) = \Psi(\alpha(x))$. If

$$(4.3) \qquad \exists\alpha_1, \alpha_2 \quad 0 \le \alpha_1 < a_2, \text{ such that } \frac{d\tilde{\Psi}}{dx}(\alpha_1) \le 0,\ \frac{d\tilde{\Psi}}{dx}(\alpha_2) \ge 0,$$

then, by continuity of $\tilde{\Psi}'_x$ there exists $\hat{\alpha} \in [\alpha_1, \alpha_2]$ such that $\frac{d\tilde{\Psi}}{dx}(\hat{\alpha}) = 0$ and $\frac{d^2\tilde{\Psi}}{dx^2}(\hat{\alpha}) > 0$. This means also that $\frac{d\Psi}{d\alpha}(\hat{\alpha}) = 0$ and $\frac{d^2\Psi}{d\alpha^2}(\hat{\alpha}) > 0$, i.e., Ψ attains a local minimum at $\hat{\alpha}$. If $f_\perp \neq 0$ then the above condition becomes simpler, since

$$\frac{d\tilde{\Psi}}{dx} = y - \sum_{i=1}^{r} \frac{\alpha^2|f_i|^2}{(\sigma_i^2 + \alpha^2)^2} - \frac{1}{\alpha^2}\|f_\perp\|^2 \longrightarrow -\infty, \text{ as } \alpha \to 0,$$

and we can put $\alpha_1 = 0$.

Since $\frac{d\tilde{\Psi}}{dx} = y + x\frac{dy}{dx}$ and $\frac{dy}{dx} = -\frac{1}{\alpha^2}$ (cf. (2.12)), conditions (4.3) may be rewritten in the form (4.1). □

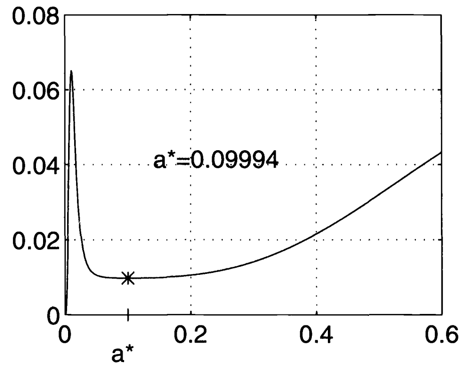


FIG 3. *The function $\psi(\alpha)$. $a* = \arg\min_{\sigma_1<\alpha<\sigma_2} \psi(\alpha)$.*

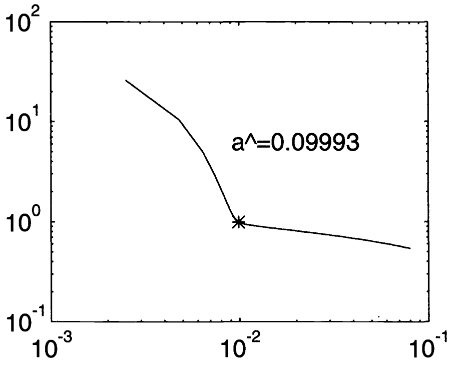


FIG 4. *The $L_{\log}$-curve. Maximum of the L-curve curvature for α =a^.*

Now let us introduce a class $\mathcal{K}$ of discrete ill-posed problems $Ku = f$. Let $\sigma_{i_1} = \sigma_1 > \sigma_{i_2} > \cdots > \sigma_{i_s} = \sigma_r$ be all different singular values of K greater than 0 with the multiplicities m_{i_j}, $j = 1, \dots, s$ ($s \geq 2$), respectively. Let u_i, $i = 1, \dots, r$, denote the corresponding left singular vectors. The right-hand side then has the representation $f = \sum_{i=1}^{r} f_i u_i + f_\perp$, where $f_i = u_i^* f$ and $f_\perp \perp u_i$ for $i = 1, \dots, r$. Let $g_{i_j}^2 := \sum_{k=0}^{m_{i_j}-1} |f_{i_j+k}|^2$.

The problem $Ku = f$ belongs to a class $\mathcal{K}$ iff $f_\perp \neq 0$ or $f_\perp = 0$ and the following conditions are satisfied:

(i) $\sigma_{i_j} \leq \frac{1}{2}\sigma_{i_{j-1}}$ for $j = 2, \dots, s$, and

(ii) $\frac{g_{i_j}^2}{\sigma_{i_j}} > 2.5\frac{g_{i_{j-1}}}{\sigma_{i_{j-1}}}$ for $j = 2, \dots, s$.

THEOREM 2. *Let the problem $Ku = f$ belong to the class $\mathcal{K}$.*

(i) *If $f_\perp = 0$ and $\sqrt{3} g_{i_s} \leq g_{i_j}$ for $j = 1, \dots, s-1$ then there exists a local minimum of $\Psi(\alpha)$ inside the open interval $(0, \infty)$.*

(ii) *If $f_\perp \neq 0$ then there exists λ such that $\Psi_\lambda(\alpha)$ has a local minimum inside the open interval $(0, \infty)$. This local minimum exists for $\lambda = 1$ if $\|f_\perp\|$ is sufficiently small.*

The proof is based on Lemma 6. The parameters α_1, α_2 are chosen a priori. A detailed proof of the first part is in [15]. The second part follows from the equivalence of the L-curve corner criterion and a local minimum of Ψ_λ described in Remark 3 and from Lemma 3 for $f_\perp \neq 0$.

CONCLUSION 1. *From Theorem 2 and Remark 3 it follows that if the problem* (1.1) *belongs to the class $\mathcal{K}$ then there exists an L-corner of the curve $L_{\ln}$.*

CONCLUSION 2. *Let $\sigma_{i-1} > 2\sigma_i > 0$ for $i = 2, \dots, r$, and let*

$$f \in \{f \in \mathbb{C}^m : |f_i| \geq \sqrt{3}|f_r| \text{ and } \|f_\perp\| \leq \gamma\}.$$

If there are no local minima of $\Psi(\alpha)$ inside $(0, \infty)$ then the $|f_i|^2$ tend to zero faster than the corresponding σ_i. In such a case it is reasonable to take $\alpha = 0$ assuming that the right-hand side is unperturbed.

Let us return to the example considered in §2, when the decomposition of the right-hand side in the basis $\{u_i\}_{i=1}^r$ has two nonzero components only: $f_1 = u_1^* f$ and $f_2 = u_2^* f$, where u_1, u_2 correspond to the singular values σ_1 and σ_2, respectively. Figures 3 and 4 present the plot of the function ψ in a neighbourhood of the smaller singular value σ_2 and the plot of the corresponding $L_{\log}$-curve for the case $\sigma_1 = 1, \sigma_2 = 0.01, f_1 = 1, f_2 = 0.1$. The regularization parameters defined by the local minimum criterion and the L-corner criterion are denoted by a* and a^, respectively. Their difference a*−a^ is of the order $0.1\sigma_2^2$ in this case.

REFERENCES

[1] A. Björck, *Least squares methods*, in Handbook of Numerical Analysis, Vol. I: Finite Difference Methods - Solution of Equations in R^n, P. G. Ciarlet and J. L. Lions, eds., Elsevier, New York, 1990.

[2] A. Björck and L. Eldén, *Methods in numerical algebra for ill posed problems*, Report LiTH-MAT-R-33-1979, Dept. of Math., Linköping University.

[3] L. Eldén. *Algorithms for the computation of functionals defined on the solution of a discrete ill-posed problem*, BIT, 30 (1990), pp. 466-483.

[4] G. H. Golub, M. Heath, and G. Wahba, *Generalized cross-validation as a method for choosing a good ridge parameter*, Technometrics, 21 (1979), pp. 215-222.

[5] C. W Groetsch, *The Theory of Tikhonov Regularization for Fredholm Equations of the First Kind*, Research Notes in Mathematics 105, Pitman, Boston, 1984.

[6] P. C. Hansen, *Truncated singular value decomposition solutions to discrete ill posed problems with ill-determined numerical rank*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 503-518.

[7] P. C. Hansen and D. P. O'Leary, *The use of the L-curve in the regularization of discrete ill-posed problems*, Technical Report UNI*C, UMIACS-TR-91-142, 1991.

[8] P. C. Hansen, *Analysis of discrete ill-posed problems by means of the L-curve*, SIAM Rev., 34 (1992) pp. 561-580.

[9] C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1995.

[10] V. A. Morozov, *Methods of solving incorrectly posed problems*, Springer-Verlag, New York, 1984.

[11] M. Z. Nashed, *Approximate regularized solutions to improperly posed linear integral and operator equations*, Lecture Notes in Math. 430, 1974, pp. 289-333.

[12] T. Regińska, *A class of regularization methods for ill-posed problems with nonexact data*, Numer. Funct. Anal. Optim., 13 (1992), pp. 601-614.

[13] ———, *Approximate solving ill-posed problems*, Matematyka Stosowana XXXI, 1989, pp. 119-135. (In Polish.)

[14] ———, *Remarks on choosing a regularization parameter in Tikhonov method*, preprint 499 IMPAN, Warszawa, 1992.

[15] ———, *A regularization parameter in discrete ill-posed problems*, preprint 514 IMPAN, Warszawa, 1993.

[16] A. N. Tikhonov and V. Y. Arsenin, *Solutions of ill-posed problem*, Winston-Wiley, New York, 1977.

[17] G. M. Vainikko and A. J. Veretennikov, *Iteration procedures in ill-posed problems*, Nauka, Moscow, 1986. (In Russian.)

[18] G. Wahba, *Practical approximate solutions to linear operator equations when data are noisy*, SIAM J. Numer. Anal., 14 (1977), pp. 651-667.

SIAM J. SCI. COMPUT.
Vol. 17, No. 3, pp. 750–761, May 1996

013

COMPUTATION OF SHOT-NOISE PROBABILITY DISTRIBUTIONS AND DENSITIES*

JOHN A. GUBNER†

Abstract. The computation of the cumulative distribution (cdf), the complementary cdf (ccdf), and the density of certain shot-noise random variables is discussed. After subtracting off a few terms that can be computed in closed form, what remains can be approximated by a general method for approximating samples of a cdf or ccdf by summing a Fourier series whose coefficients are modulated samples of their characteristic function. To approximate the density, a spline is fit to the cdf samples and then differentiated. When the density has corners, it is important that the spline have coincident knots at these locations. For shot-noise densities, these locations are easily identified.

Key words. filtered point process, Poisson process

AMS subject classification. 62E17

1. Introduction.

The computation of the cumulative distribution (cdf) $F(y)$, the complementary cdf (ccdf) $F^c(y) := 1 - F(y)$, and the probability density function $f(y)$ of a shot-noise random variable

$$Y := \sum_{\nu} A_\nu g(T_\nu)$$

is considered; here the $\{T_\nu\}$ are points of a Poisson process in $\mathbb{R}^d$ with intensity function $\lambda: \mathbb{R}^d \to [0, \infty)$, and $\{A_\nu\}$ is an independent, identically distributed, nonnegative "gain" sequence. We assume $\{A_\nu\}$ and $\{T_\nu\}$ are statistically independent. The function $g: \mathbb{R}^d \to \mathbb{R}$ is typically determined by a system impulse response or point-spread function, depending on the application in which it arises; e.g., optical communications, imaging, etc. In these situations there has been much interest in computing shot-noise distributions and densities [6, 7, 9–15, 18]. One of the reasons that this is such a hard problem is that many shot-noise densities contain jumps and corners, in addition to an impulse at the origin. It is precisely these features that make it difficult to directly apply simple methods such as Fourier inversion of the characteristic function or summation of Beaulieu's series [2]. The purpose of this paper is to expose the structure of shot-noise distributions and densities and to exploit this structure to obtain easy-to-compute approximations with the proper jumps and corners in order to obtain better accuracy.

We point out that our results also have an immediate application to the computation of the distribution of the test statistic corresponding to the likelihood ratio for Poisson process observations. Suppose we are testing the hypothesis $H_0: \lambda(\tau) = \lambda_0(\tau)$ against $H_1: \lambda(\tau) = \lambda_1(\tau)$ based on observing the points $\{T_\nu\}$ themselves. Then the likelihood ratio test for this problem is equivalent to comparing the shot-noise random variable $\mathcal{L} := \sum_\nu \ell(T_\nu)$ against a threshold, where $\ell(T_\nu) := \ln(\lambda_1(T_\nu)/\lambda_0(T_\nu))$ [16, p. 94, eq. (2.112)].

The paper is organized as follows. In §2 we state our results concerning the structure of shot-noise characteristic functions, cdfs, ccdfs, and densities. This structure allows us to decompose shot-noise distributions into various terms that can be separately approximated. In §3 we discuss a general method for approximating a cdf and ccdf from their characteristic

*Received by the editors June 1, 1994; accepted for publication (in revised form) December 14, 1994. This research was presented in part at the 1993 IEEE International Symposium on Information Theory, San Antonio, TX, January 17–22, 1993. This work was supported by the Air Force Office of Scientific Research under grants AFOSR-90-0181 and F49620-92-J-0305.

†Department of Electrical and Computer Engineering, University of Wisconsin, Madison, WI 53706 (gubner@engr.wisc.edu).

function. If the distribution has a density that is piecewise smooth except for corners, and if their locations can be identified, then excellent density approximation can be obtained by fitting a spline with coincident knots to the cdf approximation and then differentiating the spline. In §4 we use this general method to approximate one of the terms in the decomposition of a shot-noise cdf, ccdf, and density. Numerical examples are given. The conclusion is in §5.

2. Statement of results and their applications. The characteristic function of Y [17, p. 221] is

$$\varphi(\omega) := \mathsf{E}[e^{j\omega Y}] = e^{\psi(\omega)},$$

where

$$\psi(\omega) := \int \lambda(\tau)[\varphi_A(\omega g(\tau)) - 1]\, d\tau, \tag{1}$$

and φ_A is the common characteristic function of the $\{A_\nu\}$. In Theorem 5 in Appendix A we show that the condition

$$B_0 := \int_{\{\tau : g(\tau)\neq 0\}} \lambda(\tau)\, d\tau \ < \infty \tag{2}$$

implies that $\psi(\omega) = \tilde{\psi}(\omega) - B$, where $B = B_0 \mathsf{P}(A_\nu > 0)$, and $\tilde{\psi}$ is the characteristic function of a finite measure Γ on $\mathbb{R}$. The reason for doing this is that we can then write $\varphi(\omega) = e^{\psi(\omega)} = e^{-B} e^{\tilde{\psi}(\omega)}$ and see that for $C \subseteq \mathbb{R}$,

$$\mathsf{P}(Y \in C) = e^{-B}\left[\mathbf{1}_C(0) + \sum_{m=1}^{\infty} \frac{\Gamma^{*m}(C)}{m!}\right], \tag{3}$$

where $\mathbf{1}$ is the indicator function of the specified set, and Γ^{*m} is the finite measure whose characteristic function is $\tilde{\psi}(\omega)^m$. If Γ has density γ, (Theorem 6 and the remark following it in Appendix A give sufficient conditions for the existence of γ), then the density of Y, denoted by f, can be written as

$$f(y) = e^{-B}\left[\delta(y) + \sum_{m=1}^{\infty} \frac{\gamma^{*m}(y)}{m!}\right], \tag{4}$$

where γ^{*m} denotes the convolution of γ with itself m times.

2.1. Standard example. In order to illustrate our results, it is useful to keep in mind a simple example to which we can refer. We take $\lambda(\tau)$ to be a positive constant, say λ_0, and let $g(\tau) = (1-\tau)\mathbf{1}_{[0,1]}(\tau)$. Take $A_\nu \equiv 1$. Then $B = B_0 = \lambda_0$, and a simple calculation shows $\psi(\omega) = \lambda_0[(e^{j\omega} - 1)/(j\omega) - 1]$ for $\omega \neq 0$, and $\psi(0) = 0$. Hence, $\tilde{\psi}(\omega) = \lambda_0(e^{j\omega} - 1)/(j\omega)$. Note that $\tilde{\psi}$ is the characteristic function of the unnormalized density $\gamma(\theta) = \lambda_0 \mathbf{1}_{[0,1]}(\theta)$.

2.2. Application of (3) and (4) to density approximation. For $M = 0, 1, \ldots,$ let

$$\mu_M(C) := \sum_{m=M+1}^{\infty} \frac{\Gamma^{*m}(C)}{m!}.$$

Then the characteristic function of μ_M is

$$R_M(\omega) := \sum_{m=M+1}^{\infty} \frac{\tilde{\psi}(\omega)^m}{m!}.$$

Suppose $\widetilde{\psi}(\omega)$ is $O(\omega^{-\alpha_0})$ for some $\alpha_0 > 0$. Then $R_M(\omega)$ is $O(\omega^{-(M+1)\alpha_0})$. Given any positive integer p, for large enough M, $(M+1)\alpha_0 > p$, and thus μ_M is absolutely continuous; in fact, $G_M(y) := \mu_M(-\infty, y]$ has p uniformly continuous derivatives [8, §§2.7 and 11.6]. If we make the additional assumption that Γ has density γ, then (4) becomes

$$f(y) = e^{-B}\left[\delta(y) + \sum_{m=1}^{M} \frac{\gamma^{*m}(y)}{m!} + \frac{d}{dy}G_M(y)\right]. \tag{5}$$

For example, if $\alpha_0 = 1$, then G_1 has one continuous derivative, and $G_1(y) = \int_{-\infty}^{y}(\gamma * \gamma)(\theta)/2\, d\theta + G_2(y)$, where G_2 has two continuous derivatives. If we know, for example, that γ is piecewise continuous, and we know the locations of its jump discontinuities, say 0 and 1 as in our standard example, then *even without explicitly computing* $\gamma * \gamma$, we see that it, and hence the density of G_1, have corners at 0, 1, and 2. Hence, it is easy to approximate G_1 by a cubic spline $\widetilde{G}_1$ with double knots at 0, 1, 2, and simple knots elsewhere [4]. We then have $f(y) \approx e^{-B}\left[\delta(y) + \gamma(y) + \frac{d}{dy}\widetilde{G}_1(y)\right]$.

To fit a spline to G_M, we require samples of the function G_M. In §3 we discuss a method to approximate samples of G_M by applying a fast Fourier transform (FFT) to modulated samples of its characteristic function. Computable error bounds are also given.

Remark. The choice of M in (5) is limited by the cost of computing the convolutions $\gamma^{*m}(y)$. For $M = 0$ or 1, no convolutions are required.

2.3. Application of (3) to the computation of tail probabilities. For $M = 1, 2, \ldots$, let

$$\eta_M(C) := \sum_{m=1}^{M} \frac{\Gamma^{*m}(C)}{m!}.$$

Take $\eta_0(C) \equiv 0$. Then (3) becomes $\mathsf{P}(Y \in C) = e^{-B}[1_C(0) + \eta_M(C) + \mu_M(C)]$. In many applications, the measure Γ has bounded support; i.e., there exist $\theta_{\min} < \theta_{\max}$ such that $\Gamma(C) = 0$ if $C \subseteq (-\infty, \theta_{\min}) \cup (\theta_{\max}, \infty)$. Since Γ^{*m} is the convolution of Γ with itself m times, each Γ^{*m}, and hence η_M, has bounded support. It follows that for C outside the support of η_M, and for $0 \notin C$, $\mathsf{P}(Y \in C) = e^{-B}\mu_M(C)$. For example, if $\theta_{\min} < 0 < \theta_{\max}$, then

$$\mathsf{P}(Y \le y) = e^{-B}G_M(y), \quad y < M\theta_{\min},$$

and setting $G_M^c(y) := \mu_M(y, \infty)$,

$$\mathsf{P}(Y > y) = e^{-B}G_M^c(y), \quad y > M\theta_{\max}.$$

In §3 we discuss a method of approximating $G_M(y)$ and $G_M^c(y)$ by summing a Fourier series whose coefficients depend on the characteristic function of G_M. Fortunately, the larger M is, the faster the characteristic function of G_M decays, reducing the number of terms to be summed from that required using the characteristic function of G_0. We give a numerical example in §4.

3. Fourier series/spline approximation for cdfs and densities. We begin with some notation. We say G is an unnormalized cdf if G is nondecreasing, right continuous, and satisfies $G(-\infty) = 0$ and $G(\infty) < \infty$. In this case we set $G^c(y) := G(\infty) - G(y)$. The characteristic function of G is

$$\Phi(\omega) := \int_{-\infty}^{\infty} e^{j\omega x}\, dG(x).$$

Set

$$(6)\quad G_{L,N}(y) := \sum_{n=-N}^{N} b_n \Phi\left(\frac{n\pi}{L}\right) e^{-jn\pi y/L} \quad \text{and} \quad G^c_{L,N}(y) := \sum_{n=-N}^{N} b_{-n} \Phi\left(\frac{n\pi}{L}\right) e^{-jn\pi y/L},$$

where

$$(7)\qquad b_n = \begin{cases} 1/2, & n = 0, \\ j/n\pi, & n \text{ odd}, \\ 0 & \text{otherwise}. \end{cases}$$

Under certain conditions, as $N \to \infty$, $G_{L,N}(y)$ and $G^c_{L,N}(y)$ converge to the infinite sums

$$(8)\qquad G_L(y) := \lim_{N\to\infty} G_{L,N}(y) \quad \text{and} \quad G^c_L(y) := \lim_{N\to\infty} G^c_{L,N}(y).$$

A sufficient condition for these limits to exist is that $\Phi(\omega)$ be $O(\omega^{-\alpha})$ for some $\alpha > 0$. In fact, the weaker condition (9) below suffices.

THEOREM 1. *Let G be an unnormalized cdf with characteristic function Φ. If*

$$(9)\qquad \lim_{L\to\infty} \frac{1}{2L} \int_{-L}^{L} |\Phi(\omega)|^2 \, d\omega = 0,$$

then the limits in (8) *exist and satisfy*

$$(10)\qquad -G(-L+y) \le G_L(y) - G(y) \le G^c(L+y),$$

and

$$(11)\qquad -G^c(L+y) \le G^c_L(y) - G^c(y) \le G(-L+y).$$

A nonrigorous derivation of (11) was first given by Beaulieu [2] under the assumption that G possessed a density. In Appendix B, our proof of Theorem 1 shows that condition (9) is sufficient. Condition (9) is important in applications where one knows only Φ and is trying to find G; i.e., one does not know a priori if G has a density.

Remarks. There are several important observations to make about (10) and (11). (Similar observations are made in [2].) Since G is an unnormalized cdf, for any fixed y, both $G(y-L)$ and $G^c(L+y)$ in (10) and (11) go to zero as $L \to \infty$; i.e.,

$$\lim_{L\to\infty} G_L(y) = G(y) \quad \text{and} \quad \lim_{L\to\infty} G^c_L(y) = G^c(y).$$

Suppose that for some $L_1 < L_2$, $G(x) = 0$ for $x \le L_1$, and $G^c(x) = 0$ for $x \ge L_2$. Then $G(y-L)$ and $G^c(L+y)$ would both be zero when $y \in [L_1, L_2]$ and $L > L_2 - L_1$; it would then follow from (10) and (11) that $G_L(y) = G(y)$ and $G^c_L(y) = G^c(y)$. This would be the situation, for example, if G were the cdf of a bounded random variable. Similarly, if one knows only that $G(x) = 0$ for $x < 0$, corresponding to a nonnegative random variable, then $G(y-L) = 0$ for $0 \le y < L$; i.e., the lower bound in (10) and the upper bound in (11) are both zero. A weaker condition than bounded support would be the existence of the moment generating function of G. In this case, numerical bounds on $G(-L+y)$ and $G^c(L+y)$ can be obtained via the Chernoff bound.

The preceding theorem and remarks address the convergence of G_L and G^c_L to G and G^c, respectively. We next address the respective convergence of $G_{L,N}$ and $G^c_{L,N}$ to G_L and G^c_L.

THEOREM 2. *Suppose that $\Phi(\omega)$ is $O(\omega^{-\alpha})$ for some $\alpha > 0$; i.e., suppose that there exist nonnegative constants K and ω_0 such that $|\Phi(\omega)| \leq K|\omega|^{-\alpha}$ for $|\omega| > \omega_0$. Then for $N > \max\{1, \omega_0 L/\pi\}$, both*

$$|G_L(y) - G_{L,N}(y)| \quad \text{and} \quad |G_L^c(y) - G_{L,N}^c(y)| \tag{12}$$

are upper bounded by

$$\frac{K}{\alpha\pi}\left[\frac{L}{(N-1)\pi}\right]^{\alpha}.$$

Proof. Observe that the differences in (12) are just the tails of the series defining $G_{L,N}$ and $G_{L,N}^c$. Since the terms of these series have the same magnitude for positive and negative n, we can upper bound either quantity in (12) by

$$2\sum_{\substack{n=N+1\\ n\text{ odd}}}^{\infty} \frac{|\Phi(n\pi/L)|}{n\pi} \leq 2\sum_{\substack{n=N+1\\ n\text{ odd}}}^{\infty} \frac{K}{n\pi}\left(\frac{L}{n\pi}\right)^{\alpha},$$

provided that $(N+1)\pi/L > \omega_0$. Now apply an integral comparison to the sum over odd $n \geq N+1$ of $1/n^{\alpha+1}$. □

Theorem 2 says that for fixed L, $G_{L,N}$ and $G_{L,N}^c$ converge uniformly in y. We now observe that (10) and (11) imply that for y restricted to a finite closed interval, say $[L_1, L_2]$, G_L and G_L^c also converge uniformly. To see this, note that for $y \in [L_1, L_2]$, $G^c(L+y) \leq G^c(L+L_1)$ and $G(y-L) \leq G(L_2-L)$. The following result is now an immediate consequence of the triangle inequality and Theorems 1 and 2.

COROLLARY 3. *If $\Phi(\omega)$ is $O(\omega^{-\alpha})$ for some $\alpha > 0$, then $G_{L,N}$ and $G_{L,N}^c$ converge uniformly to G and G^c, respectively, on any finite closed interval as L and N become large.*

3.1. Density approximation by splines.

If Φ is integrable, then G is absolutely continuous with a uniformly continuous density [3, p. 301], which we denote by G'. In particular, if $\Phi(\omega)$ is $O(\omega^{-\alpha})$ for some $\alpha > 1$, then Φ is integrable, and for fixed $L > 0$,

$$\frac{d}{dy}G_{L,N}(y) = \frac{1}{L}\sum_{\substack{n=-N\\ n\text{ odd}}}^{N} \Phi\left(\frac{n\pi}{L}\right)e^{-jn\pi y/L} \tag{13}$$

converges uniformly to $\frac{d}{dy}G_L(y)$ as $N \to \infty$. The problem is that (13) converges more slowly than (6). Hence, we propose the following approach for approximating G'.

To begin, let $c_0 := 0$ and $c_n := b_n\Phi(n\pi/L)$ for $n \neq 0$. Then c_{-n} is the complex conjugate of c_n, and $c_N = 0$ for N even. We can therefore write, for even N,

$$G_{L,N}(y) = b_0\Phi(0) + 2\,\mathrm{Re}\sum_{n=0}^{N-1} c_n e^{-jn\pi y/L}.$$

If we let $y = k\,\Delta y$, where $\Delta y = 2L/N$, then the samples $G_{L,N}(k\,\Delta y)$ for $k = 0, \ldots, N-1$ can be computed with an N-point FFT. We can then use these samples to construct a spline approximation of $G_{L,N}(y)$; to approximate G', we differentiate the spline. The advantage of approximating G' by differentiating a spline approximation of $G_{L,N}$ rather than differentiating $G_{L,N}$ itself is that the series for $G_{L,N}$ converges faster (in N) than the series for its derivative.

3.1.1. Complexity of the approximation. To estimate the complexity of the approximation, we proceed as follows. The first step, computing an FFT of N characteristic function samples to obtain N cdf samples is an order $N \log N$ operation. Next, to compute a cubic spline approximation using the N cdf samples and i interior knots is an order $2N + (i + 8)$ operation using the NAG Library routine E02BAF [19]. Hence, these two steps combined are at most an order $N \log N$ operation. To evaluate the spline or its derivative at one point is an order $\log(i + 8)$ operation using the NAG routine E02BCF [19], which is independent of the number of characteristic function samples N.

3.1.2. Other considerations. Ordinary spline fitting routines, which operate by efficiently solving a highly structured set of linear equations, do not guarantee that a spline fit to $G_{L,N}$ will be monotone or, equivalently, that the spline derivative will be nonnegative. Although we have not found this to be a serious problem in our experience, we briefly discuss three solutions. The simplest solution to the negative density problem is to replace the spline derivative by zero whenever it is negative. Since the spline derivative typically becomes negative only in regions where the true density is almost zero, zero itself is a good approximation. However, this method does yield a nonsmooth density approximation, which may not be acceptable. A second solution, which works in our examples, is simply to increase N. Finally, we note that there are algorithms for constructing monotone splines; e.g., [1] and the references therein. However, such algorithms require solving *non*linear equations.

4. Numerical examples. Motivated by (5), we consider spline approximations of the form

$$\tilde{f}_{M,L,N}(y) := e^{-B}\left[\delta(y) + \sum_{m=1}^{M} \frac{\gamma^{*m}(y)}{m!} + \frac{d}{dy}\tilde{G}_{M,L,N}(y)\right], \tag{14}$$

where $\tilde{G}_{M,L,N}$ is a spline fit to

$$G_{M,L,N}(y) := \sum_{n=-N}^{N} b_n R_M\left(\frac{n\pi}{L}\right) e^{-jn\pi y/L}. \tag{15}$$

We also consider the direct Fourier series approximation (cf. (13))

$$f_{M,L,N}(y) := e^{-B}\left[\delta(y) + \sum_{m=1}^{M} \frac{\gamma^{*m}(y)}{m!} + \frac{d}{dy}G_{M,L,N}(y)\right]. \tag{16}$$

For our standard example (§2.1), f is known in closed form, and so in Example 1 we set $M = 1$ and compare both $\tilde{f}_{M,L,N}$ and $f_{M,L,N}$ with f. In Example 2 we apply the ideas from §2.3 to compute a tail probability $\mathsf{P}(Y > y)$ for our standard example.

Example 1. For our standard example, the density of Y was obtained in closed form, modulo a typo, by Gilbert and Pollack [6, p. 344]. The correct formula is, for $y > 0$,

$$f(y) = \lambda_0 e^{-\lambda_0} \sum_{k=0}^{\lfloor y \rfloor} \frac{(-1)^k}{k!}\left(\sqrt{\lambda_0(y-k)}\right)^{k-1} I_{k-1}\left(2\sqrt{\lambda_0(y-k)}\right),$$

where $\lfloor y \rfloor$ denotes the greatest integer less than or equal to y, and I_k is the modified Bessel function of order k. Note that $I_{-1} = I_1$. A plot of $f(y)$, $y > 0$, is shown in Fig. 1 for the case $\lambda_0 = 1$. When $\lambda_0 = 1$ we approximate $f(y)$ on $[0, 7.1]$ as follows. First we must select the knots for the spline approximation of $G_{1,L,N}$ in (15). To begin, we placed 35 uniformly spaced interior knots between 0.2 and 7. The cubic spline program we used (NAG Library routine

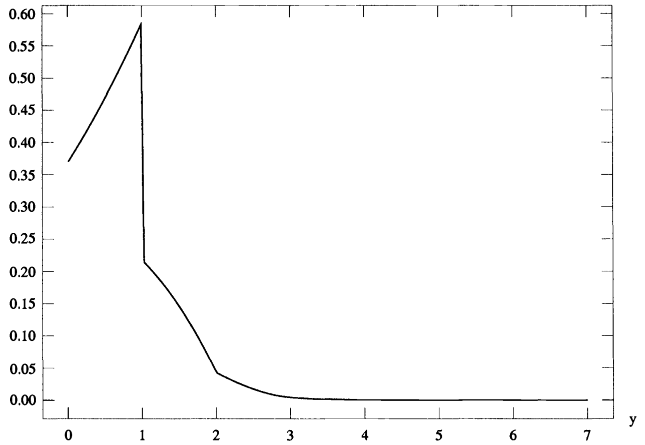


FIG. 1. *Plot of* $f(y)$ *from Example* 1.

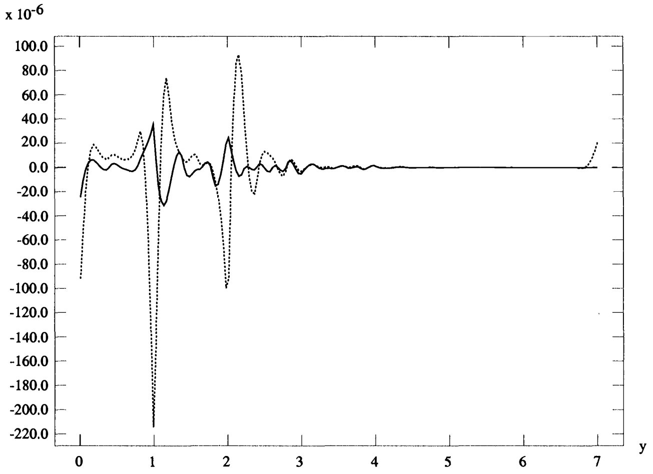


FIG. 2. *Plot of* $f(y) - \widetilde{f}_{1,L,128}(y)$ *(dotted line) and* $f(y) - \widetilde{f}_{1,L,256}(y)$ *(solid line) from Example* 1.

E02BAF) automatically put four coincident knots at each of the endpoints, 0 and 7.1. Now, as noted in §2.2, the density of G_1 has corners at 0, 1, and 2. We therefore added a coincident knot at 1 and at 2, for a total of 37 interior knots. We now take $L = 10$. If $N = 128$, a plot of $\widetilde{f}_{1,L,N}(y)$ would be graphically indistinguishable from the curve in Fig. 1. Hence, in Fig. 2 we plot the errors $f(y) - \widetilde{f}_{1,L,128}(y)$ (dotted line) and $f(y) - \widetilde{f}_{1,L,256}(y)$ (solid line) for $y \in [0.01, 7]$. Letting $\widetilde{F}_{M,L,N}(y) := e^{-B}[\mathbf{1}_{[0,\infty)}(y) + \eta_M(y) + \widetilde{G}_{M,L,N}(y)]$, we note that $\widetilde{F}_{1,L,128}(7.03125) = 1.00000$; i.e., our density approximation integrates to 1 to the number of digits shown. The spline fitting routine we used did not require that $\widetilde{f}_{1,L,N}$ be nonnegative. In fact, when $N = 128$ and y is close to 7, $\widetilde{f}_{1,L,N}(y) \approx -10^{-5}$. For $N = 256$, this problem disappears.

Finally, we note that our spline approximation (14) yields a much smaller error than the direct Fourier series approximation (16). This is demonstrated in Fig. 3 where we plot $f(y) - \widetilde{f}_{1,L,256}(y)$ (solid line) and $f(y) - f_{1,L,256}(y)$ (dotted line).

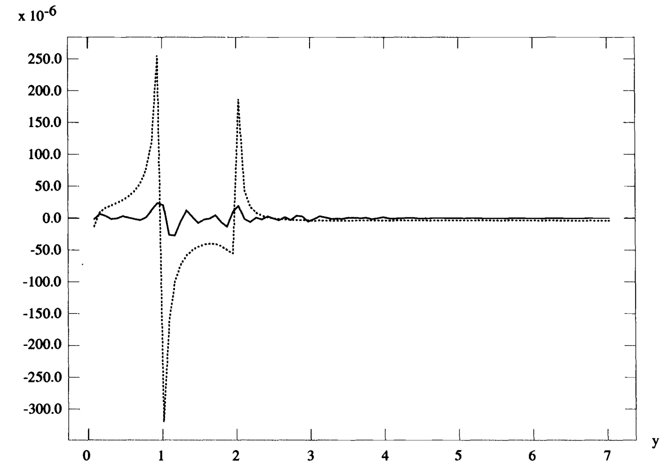


FIG. 3. *Plot of* $f(y) - \tilde{f}_{1,L,256}(y)$ *(solid line) and* $f(y) - f_{1,L,256}(y)$ *(dotted line) from Example* 1.

Example 2. As noted in §2.3, in many cases we can write $F^c(y) = e^{-B}G^c_M(y)$ for large y. Suppose $|\tilde{\psi}(\omega)| \le K_0|\omega|^{-\alpha_0}$ for large ω. Letting $\alpha = \alpha_0(M+1)$ and

$$K = \sum_{m=M+1}^{\infty} \frac{K_0^m}{m!},$$

we have $|R_M(\omega)| \le K|\omega|^{-\alpha}$. For our standard example, we have $K_0 = 2\lambda_0$ and $\alpha_0 = 1$. Hence, $R_M(\omega)$ is $O(\omega^{-(M+1)})$. Since γ lives on $[0,1]$, we see that for $y \ge M$, $\mathsf{P}(Y > y) = e^{-\lambda_0}G^c_M(y)$. By Theorem 1, the error between $\mathsf{P}(Y > y)$ and $e^{-\lambda_0}G^c_{M,L}(y)$ is less than $e^{-\lambda_0}G^c_M(L+y)$ and, by Theorem 2, the error between $e^{-\lambda_0}G^c_{M,L}(y)$ and $e^{-\lambda_0}G^c_{M,L,N}(y)$ is less than

$$(17) \qquad e^{-\lambda_0}\frac{K}{\alpha\pi}\left[\frac{L}{(N-1)\pi}\right]^{\alpha}, \quad N > 1.$$

Taking $\lambda_0 = 1$, $L = 10$, $N = 64$, and $y = M = 6$, we computed $e^{-1}G^c_{6,10,64}(6) = 1.3012036133501\times10^{-7}$. We conclude that $\mathsf{P}(Y > 6) = 1.3012\times10^{-7}$ to the number of digits shown as follows. First, using a Chernoff bound, we find numerically that $e^{-1}G^c_6(10+6) \le 8\times10^{-24}$. Second, the bound in (17) is less than 5×10^{-13}.

5. Conclusion. In §3 we discussed a general method for approximating a cdf and ccdf from their characteristic function using $G_{L,N}$ and $G^c_{L,N}$, respectively. The error due to taking L finite is given in (10) and (11); when the moment generating function exists, numerical bounds on this error can be determined with the Chernoff bound. In Theorem 2 we gave a simple bound on the error due to taking N finite. In §3.1 we proposed fitting a spline to the cdf approximation and then differentiating the spline to approximate the density. With appropriately chosen knots, excellent density approximation can be obtained.

These general results do not apply directly to shot-noise random variables because their densities contain impulses and discontinuities. Hence, Theorems 4–6 in Appendix A are the key to obtaining the decompositions (3) and (4), from which the analysis of the shot-noise density in §2.2 and the shot-noise tail probabilities in §2.3 followed. From this analysis, we see how to apply the general method of §3 to the components of the shot-noise decomposition.

In particular, the shot-noise decomposition permits the easy identification of the knot locations needed for good spline approximation.

Appendix A: Analysis of ψ. We show that $\psi(\omega) = \widetilde{\psi}(\omega) - B$, where $B = B_0\mathsf{P}(A_\nu > 0)$, and $\widetilde{\psi}$ is the characteristic function of a finite measure Γ on the Borel subsets of $\mathbb{R}$. To begin the analysis, we make the additional technical assumption that λ and g are Borel measurable. Let

$$\psi_0(\omega) := \int \lambda(\tau)[e^{j\omega g(\tau)} - 1]\,d\tau. \tag{18}$$

Then if

$$\mathsf{E}\left[\int \lambda(\tau)|e^{j\omega g(\tau)A_\nu} - 1|\,d\tau\right] < \infty, \tag{19}$$

Fubini's theorem [3, p. 200] allows (1) to be rewritten as

$$\psi(\omega) = \mathsf{E}\big[\psi_0(\omega A_\nu)\big].$$

Note that since $|e^{j\theta} - 1| \leq \min\{|\theta|, 2\}$, (2) is a sufficient condition for (19) to hold. This condition is not necessary; for example, the power-law shot-noise random variables in [9, §IV-B] satisfy (19) but not (2).

On the Borel subsets of $\mathbb{R}$, define the measure

$$\Gamma_0(C) := \int_{\{\tau : g(\tau)\in C,\, g(\tau)\neq 0\}} \lambda(\tau)\,d\tau. \tag{20}$$

Note that $\Gamma_0(\mathbb{R}) = B_0$ (cf. (2)).

Remark. The measure Γ_0 can be interpreted as an unnormalized conditional probability as follows. Recall that conditioned on the number of points $\{T_\nu\}$ occurring in a region D, the point locations in D are independent and identically distributed with common density $\lambda(\tau)/\int_D \lambda(\tau)\,d\tau$ [17, p. 90]. If $D = \{\tau \in \mathbb{R}^d : g(\tau) \neq 0\}$, then $\Gamma_0(C)/\Gamma_0(\mathbb{R})$ is the conditional probability that $g(T_\nu) \in C$ given that T_ν is one of the points in D.

THEOREM 4. *If* (2) *holds, then*

$$\psi_0(\omega) = \int_{-\infty}^{\infty} e^{j\omega\theta}\,d\Gamma_0(\theta) - \Gamma_0(\mathbb{R}).$$

Proof. Let D be a Borel subset of $\mathbb{R}^d$, and let C be a Borel subset of $\mathbb{R}$. Define two measures $\Lambda(D) := \int_D \lambda(\tau)\,d\tau$ and $H(C) := \Lambda\big(g^{-1}(C)\big)$. The change-of-variable formula for measures [3, p. 185] allows us to rewrite (18) to obtain

$$\begin{aligned}\psi_0(\omega) &= \int [e^{j\omega g(\tau)} - 1]\,d\Lambda(\tau)\\ &= \int_{-\infty}^{\infty} [e^{j\omega\theta} - 1]\,dH(\theta).\end{aligned}$$

Note that the integrand is zero for $\theta = 0$. Also note that $\Gamma_0(C) = H(C\backslash\{0\})$. Hence, we can replace $dH(\theta)$ by $d\Gamma_0(\theta)$, and the theorem follows. □

THEOREM 5. *Assume* (2) *holds, and define the finite measure*

$$\Gamma(C) := \mathsf{E}\left[\mathbf{1}_{\{A_\nu>0\}}\int_{-\infty}^{\infty} \mathbf{1}_C(A_\nu\theta)\,d\Gamma_0(\theta)\right]. \tag{21}$$

Let $B := \Gamma(\mathbb{R}) = \Gamma_0(\mathbb{R})\mathsf{P}(A_\nu > 0) = B_0\mathsf{P}(A_\nu > 0)$, *and set* $\widetilde{\psi}(\omega) := \psi(\omega) + B$. *Then*

$$\int_{-\infty}^{\infty} e^{j\omega\theta}\, d\Gamma(\theta) = \widetilde{\psi}(\omega).$$

Proof. Using the definition of Γ in (21) and proceeding in the usual way from simple functions to nonnegative functions to integrable functions, one can see that for the particular function of ζ, $e^{j\omega\zeta}$, we have

$$\int_{-\infty}^{\infty} e^{j\omega\zeta}\, d\Gamma(\zeta) = \mathsf{E}\left[\mathbf{1}_{\{A_\nu>0\}} \int_{-\infty}^{\infty} e^{j\omega(A_\nu\theta)}\, d\Gamma_0(\theta)\right],$$

which by Theorem 4 is equal to $\mathsf{E}\big[\mathbf{1}_{\{A_\nu>0\}}[\psi_0(\omega A_\nu) + \Gamma_0(\mathbb{R})]\big]$, which, since $\psi_0(0) = 0$, is equal to $\mathsf{E}\big[\psi_0(\omega A_\nu)\big] + \Gamma_0(\mathbb{R})\mathsf{P}(A_\nu > 0)$. □

THEOREM 6. *If* Γ_0 *has density* γ_0, *then* Γ *has density*

$$\gamma(\theta) = \mathsf{E}\big[\mathbf{1}_{\{A_\nu>0\}}\gamma_0(\theta/A_\nu)/A_\nu\big].$$

Proof. From the definition of Γ in (21) we can write

$$\Gamma(C) = \mathsf{E}\left[\mathbf{1}_{\{A_\nu>0\}} \int_{-\infty}^{\infty} \mathbf{1}_C(A_\nu\theta)\gamma_0(\theta)\, d\theta\right].$$

Then a change of variable followed by Tonelli's theorem [3, p. 200] yields

$$\Gamma(C) = \int_C \mathsf{E}\big[\mathbf{1}_{\{A_\nu>0\}}\gamma_0(\theta/A_\nu)/A_\nu\big]\, d\theta. \qquad \square$$

Remark. Under certain symmetry conditions, it is often quite easy to use (20) to express $\Gamma_0(\theta, \infty)$ or $\Gamma_0(-\infty, \theta]$ in a form in which γ_0 can be obtained via differentiation. A large class of such examples can be constructed as follows. Let $\|\cdot\|$ be a norm on $\mathbb{R}^d$. Let q map $[0, r)$ onto $(0, 1]$, where q is nonnegative and strictly decreasing so that q^{-1}: $(0, 1] \to [0, r)$ exists and is nonnegative and strictly decreasing. Set $g(\tau) = q(\|\tau\|)$ for $0 \le \|\tau\| < r$ and $g(\tau) = 0$ otherwise. Since $g(\tau) \ge 0$, $\Gamma_0(\theta, \infty) = B_0$ for $\theta \le 0$, and since $g(\tau) \le 1$, $\Gamma_0(\theta, \infty) = 0$ for $\theta \ge 1$. For $0 < \theta < 1$,

$$\Gamma_0(\theta, \infty) = \int_{\{\tau:\|\tau\|<q^{-1}(\theta)\}} \lambda(\tau)\, d\tau.$$

Now suppose $\lambda(\tau) = u(\|\tau\|^2)$, where u is nonnegative. If $U(\rho) = U(0) + \int_0^\rho u(x^2)x^{d-1}\, dx$, and if $\mathcal{A}_d$ denotes the area of the unit sphere $\{\tau \in \mathbb{R}^d : \|\tau\| = 1\}$, then $\Gamma_0(\theta, \infty) = \mathcal{A}_d[U\big(q^{-1}(\theta)^2\big) - U(0)]$. If q is piecewise continuously differentiable, then $\gamma_0(\theta) = -\mathcal{A}_d u\big(q^{-1}(\theta)^2\big)q^{-1}(\theta)(d/d\theta)q^{-1}(\theta)$, where the discontinuities and corners of γ_0 are easily determined from those of u and q.

Appendix B: Proof of Theorem 1. We claim that it suffices to prove the theorem for $y = 0$. To see this, suppose (10) and (11) hold with $y = 0$; note the resulting simplifications in the expressions for $G_{L,N}(0)$ and $G^c_{L,N}(0)$, in terms of which $G_L(0)$ and $G^c_L(0)$ are defined. Then replace G by G_y, where $G_y(x) := G(x + y)$, and note that the characteristic function of G_y is $\int_{-\infty}^{\infty} e^{j\omega x} dG_y(x) = \Phi(\omega)e^{-j\omega y}$. This establishes the claim.

We now prove the theorem when $y = 0$. We begin with (11). Observe that $G^c(0) = \int_{-\infty}^{\infty} \mathbf{1}_{(0,\infty)}(x)\, dG(x)$. Now, for any $L > 0$, define the periodic pulse train β_L with period $2L$ by specifying its values on $(-L, L]$ to be

$$\beta_L(x) = \begin{cases} 1, & 0 < x \le L, \\ 0, & -L < x \le 0. \end{cases}$$

Observe that for all x,

$$\beta_L(x) \leq \mathbf{1}_{(0,\infty)}(x) + \mathbf{1}_{(-\infty,-L]}(x)$$

and

$$\beta_L(x) \geq \mathbf{1}_{(0,L]}(x) = \mathbf{1}_{(0,\infty)}(x) - \mathbf{1}_{(L,\infty)}(x).$$

Hence, we have the error bounds

$$-G^c(L) \leq \int_{-\infty}^{\infty} \beta_L(x)\,dG(x) - G^c(0) \leq G(-L).$$

The next step is to approximate β_L by its Fourier series. Set

$$\beta_{L,N}(x) := \sum_{n=-N}^{N} b_{-n}e^{jn\pi x/L},$$

where the b_n are given by (7). Now, as $N \to \infty$, $\beta_{L,N}(x) \to \beta_L(x)$ for all x except at points of discontinuity of β_L. Since (9) is equivalent to the condition that the measure induced by G have no point masses [3, p. 306], $\beta_{L,N}$ converges to β_L almost everywhere with respect to this measure. Since $\beta_{L,N}(x)$ is uniformly bounded in N and x (with L fixed) [5, p. 151, eq. (10.1.5)], the dominated convergence theorem yields

$$\begin{aligned}\int_{-\infty}^{\infty} \beta_L(x)\,dG(x) &= \sum_{n=-\infty}^{\infty} b_{-n} \int_{-\infty}^{\infty} e^{jn\pi x/L}\,dG(x)\\ &= \sum_{n=-\infty}^{\infty} b_{-n}\Phi\left(\frac{n\pi}{L}\right)\\ &= G_L^c(0).\end{aligned}$$

This establishes (11). To establish (10), repeat the preceding derivation, but replace $\beta_L(x)$ by $\beta_L(-x)$. Then note that G is real and equal to its complex conjugate. □

REFERENCES

[1] L.-E. Andersson and T. Elfving, *Interpolation and approximation by monotone cubic splines*, J. Approx. Theory, 66 (1991), pp. 302-333.

[2] N. C. Beaulieu, *An infinite series for the computation of the complementary probability distribution function of a sum of independent random variables and its application to the sum of Rayleigh random variables*, IEEE Trans. Comm., 38 (1990), pp. 1463-1474.

[3] P. Billingsley, *Probability and Measure*, Wiley, New York, 1979.

[4] C. de Boor, *A Practical Guide to Splines*, Springer-Verlag, New York, 1978.

[5] R. E. Edwards, *Fourier Series, A Modern Introduction, Volume* 1, Holt, Rinehart and Winston, New York, 1967.

[6] E. N. Gilbert and H. O. Pollak, *Amplitude distribution of shot noise*, Bell Syst. Tech. J., 39 (1960), pp. 333-350.

[7] C. W. Helstrom and C.-L. Ho, *Analysis of avalanche diode receivers by saddlepoint integration*, IEEE Trans. Comm., 40 (1992), pp. 1327-1338.

[8] T. Kawata, *Fourier Analysis in Probability Theory*, Academic Press, New York, 1972.

[9] S. B. Lowen and M. C. Teich, *Power-law shot noise*, IEEE Trans. Inform. Theory, 36 (1990), pp. 1302-1318.

[10] J. E. Mazo and J. Salz, *On optical data communication via direct detection of light pulses*, Bell Syst. Tech. J., 55 (1976), pp. 347-369.

[11] G. M. Morris, *Scene matching using photon-limited images*, J. Opt. Soc. Amer. A, 1 (1984), pp. 482-488.

[12] A. Papoulis, *High density shot noise and Gaussianity*, J. Appl. Probab., 8 (1971), pp. 118-127.

[13] S. O. Rice, *Mathematical analysis of random noise*, Bell Syst. Tech. J., 23 (1944), pp. 282-332.

[14] ———, *Mathematical analysis of random noise*, Bell Syst. Tech. J., 24 (1945), pp. 46-156.

[15] W. J. Richter Jr. and T. I. Smits, *Numerical evaluation of Rice's integral representation of the probability density function for Poisson impulse noise*, J. Acoust. Soc. Amer., 56 (1974), pp. 481-496.

[16] D. L. Snyder, *Random Point Processes*, Wiley, New York, 1975.

[17] D. L. Snyder and M. I. Miller, *Random Point Processes in Time and Space*, 2nd ed., Springer-Verlag, New York, 1991.

[18] O.-C. Yue, R. Lugannani, and S. O. Rice, *Series approximations for the amplitude distribution and density of shot processes*, IEEE Trans. Comm., 26 (1978), pp. 45-54.

[19] *The NAG Fortran Library Manual, Mark* 15, *Volume* 3, The Numerical Algorithms Group Limited, Oxford, U.K., 1991.

SIAM J. Sci. Comput.
Vol. 17, No. 3, pp. 762-772, May 1996

TOEPLITZ-CIRCULANT PRECONDITIONERS FOR TOEPLITZ SYSTEMS AND THEIR APPLICATIONS TO QUEUEING NETWORKS WITH BATCH ARRIVALS*

RAYMOND H. CHAN† AND WAI-KI CHING‡

Abstract. The preconditioned conjugate gradient method is employed to solve Toeplitz systems $T_n \mathbf{x} = \mathbf{b}$ where the generating functions of the n-by-n Toeplitz matrices T_n are functions with zeros. In this case, circulant preconditioners are known to give poor convergence, whereas band-Toeplitz preconditioners offer only linear convergence and can handle only real-valued functions with zeros of even orders. We propose here preconditioners which are products of band-Toeplitz matrices and circulant matrices. The band-Toeplitz matrices are used to cope with the zeros of the given generating function and the circulant matrices are used to speed up the convergence rate of the algorithm. Our preconditioner can handle complex-valued functions with zeros of arbitrary orders. We prove that the preconditioned Toeplitz matrices have singular values clustered around 1 for large n. We apply our preconditioners to solve the stationary probability distribution vectors of Markovian queueing models with batch arrivals. We show that if the number of servers is fixed independent of the queue size n, then the preconditioners are invertible and the preconditioned matrices have singular values clustered around 1 for large n. Numerical results are given to illustrate the fast convergence of our methods.

Key words. preconditioning, Toeplitz matrix, circulant matrix, queueing network

AMS subject classifications. 65F10, 65F15, 65N20, 60K25

1. Introduction. In this paper, we discuss the solutions of linear systems $T\mathbf{x} = \mathbf{b}$ where T is a Toeplitz matrix. Direct methods that are based on the Levinson recursion formula are in constant use; see for instance, Trench [25]. For an n-by-n Toeplitz matrix T_n, these methods require $O(n^2)$ operations. Faster algorithms that require $O(n \log^2 n)$ operations have been developed; see Ammar and Gragg [1] for instance. The stability properties of these direct methods for symmetric positive definite matrices are discussed in Bunch [5].

Here we will consider solving Toeplitz systems by the preconditioned conjugate gradient squared (PCGS) method. There are many circulant preconditioning strategies for Toeplitz systems; see for instance [23, 11, 16, 15, 17, 18, 4, 3]. The convergence results for these circulant preconditioners are all based on the regularity of the function $g(\theta)$ whose Fourier coefficients give the diagonals of T_n. The function $g(\theta)$ with $\theta \in [-\pi, \pi]$ is called the generating function of the sequence of Toeplitz matrices T_n. A general result is that if $g(\theta)$ is a positive function in the Wiener class, then for large enough n, the preconditioned matrix has eigenvalues clustered around 1. In particular, the PCGS method applied to the preconditioned system converges superlinearly and the n-by-n Toeplitz system can be solved in $O(n \log n)$ operations. However, we remark that if $g(\theta)$ has a zero, then the result fails to hold and the circulant preconditioned systems can converge at a very slow rate (see the numerical examples in §4 or in [9]).

To this end, Chan in [6] used trigonometric functions of the form $\sin^\ell(\theta - \theta_0)$ to approximate the function $g(\theta)$ around the zeros θ_0 of g. The power ℓ is the order of the zero θ_0 and is required to be an even number. The resulting preconditioner is a band-Toeplitz matrix which gives linear convergence. The bandwidth of the preconditioner is $(\ell + 1)$. To speed up the convergence rate, Chan and Tang [9] have considered using the Remez algorithm to find the

*Received by the editors April 21, 1994; accepted for publication (in revised form) December 5, 1994.

†Department of Mathematics, Chinese University of Hong Kong, Shatin, Hong Kong (rchan@math.cuhk.hk). The research of this author was supported by Hong Kong Research Grant Council grants CUHK 178/93E and CUHK-DAG 220600680.

‡Department of Applied Mathematics, Polytechnic University of Hong Kong, Hung Hom, Hong Kong and Department of Systems Engineering and Engineering Management, Chinese University of Hong Kong, Shatin, Hong Kong.

best trigonometric polynomial that approximates g in the supremum norm and yet matches the order of the zeros of g in a neighborhood of the zeros. The resulting band-Toeplitz preconditioner can significantly reduce the condition number of the preconditioned systems at the expense of enlarging the bandwidth. We note that both methods work only for real-valued generating functions with zeros of even orders and fail for complex-valued functions or real-valued functions with zeros of odd order. A typical example is the Toeplitz matrix tridiag$[-1, 1, 0]$. Its generating function is given by $g(\theta) = 1 - e^{-i\theta}$ and it has a zero of order 1 at $\theta = 0$. We note that if we write $z = e^{i\theta}$, then g as a function of z has a zero of order 1 at $z = 1$.

In this paper, we will design preconditioners that give superlinear convergence and work for generating functions that are complex valued and have zeros with arbitrary orders. Our idea is to approximate g, as a function of the complex variable z, around its zeros z_0 by functions of the form $(z - z_0)^\ell$, where ℓ is the order of the zero z_0. Then we approximate the quotient $g(z)/(z - z_0)^\ell$ by using the usual circulant approach. This results in a preconditioner which is a product of a band-Toeplitz matrix with bandwidth $(\ell + 1)$ and a circulant matrix. We will prove that if the quotient is a nonvanishing Wiener-class function, then the preconditioner is invertible and the iterative method converges superlinearly for large n.

We then apply our preconditioner to solve the stationary probability distribution vectors for Markovian queueing networks with batch arrivals. We note that the generator matrices A_n for these queueing networks are singular matrices with a Toeplitz-like structure. In fact, when there is only one server in the system, A_n differs from a lower Hessenberg Toeplitz matrix by a rank-one matrix. The preconditioner P_n is constructed by exploiting the near-Toeplitz structure of A_n and will also be a product of a band-Toeplitz matrix and a circulant matrix. We prove that if the number of servers is independent of the queue size n, then for all sufficiently large n, P_n are invertible and the preconditioned matrices have singular values clustered around 1.

The outline of this paper is as follows. In §2 we define our preconditioners P_n for general Toeplitz matrices with generating functions that have zeros. We then prove that the preconditioned systems have singular values clustered around 1. In §3 we consider solving Markovian queueing networks with batch arrivals by using our preconditioners. In §4 numerical results are given to illustrate the fast convergence of our methods when compared to other methods and other preconditioners used in solving Toeplitz systems and queueing networks. Finally, concluding remarks are given in §5.

2. Construction and analysis of preconditioners.

In this section, we discuss how to construct preconditioners for Toeplitz systems T_n whose generating functions are functions having zeros. Then we analyze the convergence rate of the resulting preconditioned systems.

Let us first recall the definitions of Toeplitz and circulant matrices. An n-by-n matrix $T_n = (t_{i,j})$ is said to be Toeplitz if $t_{i,j} = t_{i-j}$, i.e., if T_n is constant along its diagonals. It is said to be circulant if its diagonals t_k further satisfy $t_{n-k} = t_{-k}$ for $0 < k \leq n - 1$. The idea of using circulant matrices as preconditioners for Toeplitz matrices has been studied extensively in recent years; see for instance [23, 11, 16, 26, 15]. In this paper, we will concentrate on the T. Chan circulant preconditioners. The results for the other circulant preconditioners can be obtained similarly; see §5.

For a given Toeplitz matrix T_n with diagonals $\{t_j\}_{j=-(n-1)}^{n-1}$, the T. Chan circulant preconditioner to T_n is defined to be the circulant matrix C_n which minimizes the Frobenius norm $||T_n - C_n||_F$ among all circulant matrices. The (i, j)th entry of C_n is given by c_{i-j}, where

$$(1) \qquad c_k = \begin{cases} \dfrac{(n-k)t_k + kt_{k-n}}{n}, & 0 \leq k < n, \\ c_{n+k}, & 0 < -k < n; \end{cases}$$

see T. Chan [11]. We note that the diagonals $\{c_j\}_{j=-(n-1)}^{n-1}$ and hence the matrix C_n can be obtained in $O(n)$ operations. The eigenvalues of C_n, which are required in the inversion of C_n, can be computed in $O(n \log n)$ operations by using fast Fourier transforms; see Strang [23] for instance. Because of the good approximating properties of the T. Chan circulant preconditioners, they have been used when solving numerical elliptic partial differential equations [7] and signal processing problems [8].

We will analyze the convergence rate of the preconditioned systems $C_n^{-1}T_n$ in the limit $n \to \infty$, assuming that a fixed sequence of entries $\{t_j\}_{j=-\infty}^{\infty}$ has been prescribed. As usual in the study of Toeplitz matrices and operators (see for instance Grenander and Szegö [14]) we consider the Laurent series

$$g(z) = \sum_{j=-\infty}^{\infty} t_j z^j$$

whose coefficients $\{t_j\}$ are the entries of T_n, with $t_{j,k} = t_{j-k}$ for $0 \le j, k < n$. We will call $g(z)$ the generating function of the sequence of Toeplitz matrices T_n and for clarity, we will denote such T_n by $\mathcal{T}_n[g]$ and the corresponding T. Chan circulant preconditioner by $\mathcal{C}_n[g]$.

We note that if

$$\sum_{j=-\infty}^{\infty} |t_j| < \infty,$$

i.e., if $g(z)$ belongs to the Wiener class of functions $\mathcal{W}$ defined on the unit circle $|z| = 1$ and if $g(z)$ has no zeros on $|z| = 1$, then $\mathcal{C}_n[g]$ is a good approximation to $\mathcal{T}_n[g]$ as far as PCGS methods are concerned.

LEMMA 2.1. *Let $g \in \mathcal{W}$ and have no zeros on $|z| = 1$. Then for large n, $\mathcal{C}_n[g]$ will be invertible and the sequence of matrices $\mathcal{C}_n[g]^{-1}\mathcal{T}_n[g]$ will have singular values clustered around* 1. *More precisely, for any fixed $\epsilon > 0$, there exist integers $M, N > 0$ such that for all $n > N$, $\mathcal{C}_n[g]$ is invertible and the matrix $\mathcal{C}_n[g]^{-1}\mathcal{T}_n[g]$ has no more than M singular values lying outside the interval* $(1-\epsilon, 1+\epsilon)$.

Proof. We note that by the Weierstrass M-test, $g(z)$ is a 2π-periodic complex-valued function defined on the unit circle $|z| = 1$ with respect to the angle θ; see for instance Conway [12, p. 29]. The lemma now follows from Lemma 3 and Theorem 2 of Chan and Yeung [10]. □

In this paper, we relax the requirement that $g(z)$ has no zeros on $|z| = 1$. In particular, we consider $g(z)$ that are of the form

$$g(z) = \left\{\prod_j (z - z_j)^{\ell_j}\right\} h(z),$$

where z_j are the roots of $g(z)$ on $|z| = 1$ with order ℓ_j and $h(z)$ is a nonvanishing function in $\mathcal{W}$. Our Toeplitz-circulant preconditioner P_n is defined to be

$$P_n = \mathcal{T}_n\left[\prod_j (z - z_j)^{\ell_j}\right] \mathcal{C}_n[h].$$

By expanding the product $\prod_j (z - z_j)^{\ell_j}$ we see that the Toeplitz matrix $\mathcal{T}_n[\prod_j (z - z_j)^{\ell_j}]$ is a lower triangular matrix with bandwidth equal to $(\ell + 1)$, where

$$\ell = \sum_j \ell_j.$$

Moreover, its main diagonal entry is 1 and therefore it is invertible for all n.

In each iteration of the PCGS method, we have to solve a linear system of the form $P_n\mathbf{y} = \mathbf{r}$. We first claim that P_n is invertible for large n. As mentioned above, the Toeplitz matrix $\mathcal{T}_n[\prod_j (z - z_j)^{\ell_j}]$ is invertible for all n. Since $h \in \mathcal{W}$ and has no zeros, the invertibility of $\mathcal{C}_n[h]$ for large n is guaranteed by Lemma 2.1. Hence P_n is invertible for large n. Let us consider the cost of solving the system

$$P_n\mathbf{y} = \mathcal{T}_n\left[\prod_j (z - z_j)^{\ell_j}\right]\mathcal{C}_n[h]\mathbf{y} = \mathbf{r}.$$

Because the matrix $\mathcal{T}_n[\prod_j (z - z_j)^{\ell_j}]$ is a lower triangular matrix with bandwidth $(\ell + 1)$, the system involving $\mathcal{T}_n[\prod_j (z - z_j)^{\ell_j}]$ can be solved by forward substitution and the cost is $O(\ell n)$ operations. Given any vector $\mathbf{x}$, the matrix-vector product $\mathcal{C}_n[h]^{-1}\mathbf{x}$ can be done by using fast Fourier transforms in $O(n \log n)$ operations; see Strang [23] and O'Leary and Simmons [20]. Thus the system $P_n\mathbf{y} = \mathbf{r}$ can be solved in $O(n \log n) + O(\ell n)$ operations. In comparison, the systems involving the preconditioners proposed by Chan [6] and Chan and Tang [9] require $O(n \log n) + O(\ell^2 n)$ operations to be solved.

We now investigate the convergence rate of the preconditioned systems.

THEOREM 2.2. *The sequence of matrices $P_n^{-1}\mathcal{T}_n[g]$ has singular values clustered around 1 for all sufficiently large n.*

Proof. Since $\mathcal{T}_n[\prod_j (z - z_j)^{\ell_j}]$ is a lower triangular Toeplitz matrix of bandwidth $(\ell + 1)$, we see that the matrix

$$\left\{\mathcal{T}_n[g] - \mathcal{T}_n\left[\prod_j (z - z_j)^{\ell_j}\right]\mathcal{T}_n[h]\right\}$$

only has nonzero entries in the first $l + 1$ rows. Hence it is clear that

$$\mathcal{T}_n[g] = \mathcal{T}_n\left[\prod_j (z - z_j)^{\ell_j}\right]\mathcal{T}_n[h] + L_1,$$

where rank $L_1 \le \ell + 1$. Therefore

$$\begin{aligned}
P_n^{-1}\mathcal{T}_n[g] &= \mathcal{C}_n[h]^{-1}\mathcal{T}_n\left[\prod_j (z - z_j)^{\ell_j}\right]^{-1}\mathcal{T}_n[g] \\
&= \mathcal{C}_n[h]^{-1}\mathcal{T}_n\left[\prod_j (z - z_j)^{\ell_j}\right]^{-1}\left(\mathcal{T}_n\left[\prod_j (z - z_j)^{\ell_j}\right]\mathcal{T}_n[h] + L_1\right) \\
&= \mathcal{C}_n[h]^{-1}\mathcal{T}_n[h] + L_2, \qquad (2)
\end{aligned}$$

where rank $L_2 \le \ell$. Since h has no zeros, by Lemma 2.1, $\mathcal{C}_n[h]^{-1}\mathcal{T}_n[h]$ has clustered singular values. In particular, we can write $\mathcal{C}_n[h]^{-1}\mathcal{T}_n[h] = I + L_3 + U$, where U is a small norm matrix and rank L_3 is fixed independent of n; see [10, Cor. 1]. Hence (2) becomes

$$P_n^{-1}\mathcal{T}_n[g] = I + L_4 + U,$$

where the rank of L_4 is again fixed independent of n. By using the Cauchy interlace theorem [28, p. 103] on

$$(P_n^{-1}\mathcal{T}_n[g])^*(P_n^{-1}\mathcal{T}_n[g]) = (I + L_4 + U)^*(I + L_4 + U),$$

it is straightforward to show that $P_n^{-1}\mathcal{T}_n[g]$ has singular values clustered around 1; see [10, Thm. 2] for details. □

Accordingly, the PCGS methods will converge quickly when applied to solving the preconditioned systems; see Axelsson and Barker [2, p. 26] for instance. Numerical examples are given in §4 to illustrate this fast convergence.

3. Markovian queueing networks. In this section, we consider using the PCGS method with our Toeplitz-circulant preconditioners for solving the stationary probability distribution vectors for Markovian queueing models with batch arrivals. This kind of queueing system occurs in many applications, such as telecommunication networks [19] and loading dock models [21]. We will see that the generator matrices of these systems have a near-Toeplitz structure and our preconditioners are constructed by exploiting this fact.

Let us first introduce the following queueing parameters. Definitions of queueing theory terminologies used below can be found in Cooper [13]. The input of the queueing system will be an exogenous Poisson batch arrival process with mean batch interarrival time λ^{-1}. For $k \geq 1$, denote λ_k to be the batch arrival rate for batches with size k. We note that

$$\lambda_k = \lambda p_k, \tag{3}$$

where p_k is the probability that the arrival batch size is k. Clearly we have

$$\sum_{k=1}^{\infty} \lambda_k = \lambda. \tag{4}$$

The number of servers in the queueing system will be denoted by s. The service time of each server is independent of the others and is exponentially distributed with mean μ^{-1}. The waiting room is of size $(n-s-1)$ and the queueing discipline is blocked-customers-cleared. If the arrival batch size is larger than the number of waiting places left, then only part of the arrival batch will be accepted; the other customers will be treated as overflows and will be cleared from the system.

By ordering the state-space lexicographically, i.e., the ith variable corresponds to the state where there are $i-1$ customers in the system, the queueing model can be characterized by the infinitesimal generator matrix

$$A_n = \begin{pmatrix} \lambda & -\mu & 0 & 0 & 0 & \dots & 0 \\ -\lambda_1 & \lambda+\mu & -2\mu & 0 & 0 & \dots & 0 \\ -\lambda_2 & -\lambda_1 & \lambda+2\mu & \ddots & \ddots & & \vdots \\ \vdots & -\lambda_2 & \ddots & \ddots & -s\mu & \ddots & \\ & \vdots & \ddots & \ddots & \lambda+s\mu & \ddots & 0 \\ -\lambda_{n-2} & -\lambda_{n-3} & & & \vdots & \lambda+s\mu & -s\mu \\ -r_1 & -r_2 & -r_3 & \cdots & -r_{s+1} & \cdots & s\mu \end{pmatrix}, \tag{5}$$

where r_i are such that each column sum of A_n is zero; see Seila [21].

Clearly A_n has zero column sum, positive diagonal entries and nonpositive off-diagonal entries. Moreover the matrix A_n is irreducible. In fact, if $\lambda_i = 0$ for all $i = 1, \dots, n-2$, then $r_1 = \lambda$ and the matrix is irreducible. If the λ_i's are not all zero, say λ_j is the first nonzero λ_i, then $r_{n-j} = \lambda$, and hence A_n is also irreducible. From the theory of Perron and Frobenius [27, p. 30], A_n has a 1-dimensional null-space with a positive null vector.

The stationary probability distribution vector $\mathbf{p}$ of the queueing system is the normalized null-vector of the generator matrix A_n given in (5). Much useful information about the queueing system, such as the blocking probability and the expected waiting time of customers, can be obtained from $\mathbf{p}$. Since A_n has a 1-dimensional null-space, $\mathbf{p}$ can be found by deleting the last column and the last row of A_n and solving the $(n-1)$-by-$(n-1)$ reduced linear system $Q_{n-1}\mathbf{y} = (0, \ldots, 0, s\mu)^t$. After getting $\mathbf{y}$, the distribution vector $\mathbf{p}$ can then be obtained by normalizing the vector $(\mathbf{y}^t, 1)^t$.

Thus let us concentrate on solving nonhomogeneous systems of the form

$$Q_n\mathbf{x} = \mathbf{b}, \tag{6}$$

where

$$Q_n = \begin{pmatrix} \lambda & -\mu & 0 & 0 & 0 & \ldots & 0 \\ -\lambda_1 & \lambda+\mu & -2\mu & 0 & 0 & \ldots & 0 \\ -\lambda_2 & -\lambda_1 & \lambda+2\mu & \ddots & \ddots & & \vdots \\ \vdots & -\lambda_2 & \ddots & \ddots & -s\mu & \ddots & \\ & \vdots & \ddots & \ddots & \lambda+s\mu & \ddots & 0 \\ -\lambda_{n-2} & -\lambda_{n-3} & & \ddots & \ddots & \ddots & -s\mu \\ -\lambda_{n-1} & -\lambda_{n-2} & & \cdots & -\lambda_2 & -\lambda_1 & \lambda+s\mu \end{pmatrix}. \tag{7}$$

Notice that if all of the λ_i, $i = 1, \ldots, n-1$ are zeros, then Q_n will be a bidiagonal matrix and can easily be inverted. Therefore in the following, we assume that at least one of the λ_i is nonzero. Then clearly Q_n^t is an irreducibly diagonally dominant matrix. In particular, if the system (6) is solved by classical iterative methods such as the Jacobi or the Gauss-Seidel method, both methods will converge for arbitrary initial guesses; see for instance Varga [27, Thm. 3.4].

We will see in §3.2 that the costs per iteration of the Jacobi and the Gauss-Seidel methods are $O(n \log n)$ and $O(n^2)$, respectively. The memory requirement is $O(n)$ for both methods. We note that the system (6) can also be solved by Gaussian elimination in $O(n^2)$ operations with $O(n^2)$ memory. In the remainder of this section, we are interested in solving (6) by the PCGS method. We will see that the cost per iteration of the method is $O(n \log n)$ and memory requirement is $O(n)$, the same as those of the Jacobi method. However, we are able to show that if s is independent of n, then with our Toeplitz-circulant preconditioner, the PCGS method converges superlinearly for all sufficiently large n. In particular, the method converges in a finite number of steps independent of the queue size n. Therefore the total cost of finding the steady-state probability distribution is $O(n \log n)$ operations.

3.1. Construction of the preconditioner. We observe that in the single server case, i.e., when $s = 1$, the matrix Q_n given in (7) differs from a lower Hessenberg Toeplitz matrix by only its $(1, 1)$ entry. In general, Q_n can be written as

$$Q_n = T_n + R_n, \tag{8}$$

where T_n is the Toeplitz matrix

$$T_n = \begin{pmatrix} \lambda+s\mu & -s\mu & 0 & 0 & 0 & \ldots & 0 \\ -\lambda_1 & \lambda+s\mu & -s\mu & 0 & 0 & \ldots & 0 \\ -\lambda_2 & -\lambda_1 & \lambda+s\mu & \ddots & \ddots & & \vdots \\ \vdots & -\lambda_2 & \ddots & \ddots & -s\mu & \ddots & \\ & \vdots & \ddots & \ddots & \lambda+s\mu & \ddots & 0 \\ -\lambda_{n-2} & & & \ddots & \ddots & \ddots & -s\mu \\ -\lambda_{n-1} & -\lambda_{n-2} & & \ldots & -\lambda_2 & -\lambda_1 & \lambda+s\mu \end{pmatrix} \tag{9}$$

and R_n is a matrix of rank s. From (9), we see that $T_n = \mathcal{T}_n[g]$, where the generating function $g(z)$ of T_n is given by

$$g(z) = -s\mu\frac{1}{z} + \lambda + s\mu - \sum_{k=1}^{\infty} \lambda_k z^k. \tag{10}$$

We note that by (4), $g \in \mathcal{W}$.

Unfortunately, it is also clear from (10) and (4) that $g(z)$ has a zero at $z = 1$ and therefore Lemma 2.1 is not applicable. However, if we look at the real part of $g(z)$ on the unit circle $|z| = 1$, we see that

$$\text{Re}\{g(z)\} = -s\mu\cos\theta + \lambda + s\mu - \sum_{k=1}^{\infty} \lambda_k \cos(k\theta) \geq s\mu - s\mu\cos\theta.$$

Hence the zeros of $g(z)$ can only occur at $z = 1$. In particular, we can write

$$g(z) = (z-1)^{\ell} b(z), \tag{11}$$

where ℓ is the order of the zero of $g(z)$ at $z = 1$ and $b(z)$ will have no zeros on the unit circle. According to the discussion in §2, we define our preconditioner for Q_n as

$$P_n = \mathcal{T}_n[(z-1)^{\ell}]\mathcal{C}_n[b]. \tag{12}$$

Let us consider cases where the quotient function $b(z)$ will be in $\mathcal{W}$. We first note that if the radius of convergence ρ of the power series $\sum_{k=1}^{\infty} \lambda_k z^k$ in (10) is greater than 1, then $g(z)$ and hence $b(z)$ are analytic functions in a neighborhood of $|z| = 1$; see Conway [12, p. 31]. In particular, $h(z)$ will be in $\mathcal{W}$. A formula for computing ρ is given by

$$\frac{1}{\rho} = \limsup |\lambda_j|^{1/j}; \tag{13}$$

see Conway [12, p. 31].

Next we consider the case $\ell = 1$ in more depth. By straightforward division of $g(z)$ in (10) by $(z - 1)$, we have

$$b(z) = s\mu\frac{1}{z} - \lambda - \sum_{k=1}^{\infty}\left(\lambda - \sum_{j=1}^{k} \lambda_j\right) z^k. \tag{14}$$

Therefore, by (3) and (4),

$$b(1) = s\mu - \sum_{k=0}^{\infty}\sum_{j=k+1}^{\infty} \lambda_j = s\mu - \lambda\sum_{k=0}^{\infty} kp_k = s\mu - \lambda E(B), \tag{15}$$

where $E(B)$ is the expected value of the arrival batch size. Thus if $s\mu \neq \lambda E(B)$ then $b(1) \neq 0$ and hence $\ell = 1$. Moreover, if $E(B) < \infty$, then $b \in \mathcal{W}$. Clearly from (14), the first n Laurent coefficients of $b(z)$, i.e., $\sum_{j=1}^{k} \lambda_j - \lambda$, $k = 1, 2, \ldots, n$, can be computed recursively in $O(n)$ operations. Hence by using (1), $\mathcal{C}_n[h]$ and also P_n can be constructed in $O(n)$ operations.

3.2. Convergence analysis and computation cost. In this section, we prove the fast convergence of the PCGS method and discuss its computational cost.

THEOREM 3.1. *Let $b(z)$ defined in (11) be in $\mathcal{W}$ and the number of servers s in the queue be independent of the queue size n. Then the sequence of preconditioned matrices $P_n^{-1}Q_n$ has singular values clustered around* 1 *for large n.*

Proof. By (8) and (12),

$$P_n^{-1}Q_n = \mathcal{C}_n[b]^{-1}\mathcal{T}_n[(z-1)^\ell]^{-1}(\mathcal{T}_n[g] + R_n) = \mathcal{C}_n[b]^{-1}\mathcal{T}_n[(z-1)^\ell]^{-1}\mathcal{T}_n[g] + L_5,$$

where rank $L_5 \le s$. By Theorem 2.2 and the Cauchy interlace theorem (see [28]), we see that $P_n^{-1}Q_n$ has singular values clustered around 1 for sufficiently large n. □

It follows from standard convergence theory of the PCGS method that the method will converge superlinearly and in particular in a finite number of steps independent of n.

In each iteration of the PCGS method, the main computational cost consists of solving a linear system $P_n\mathbf{y} = \mathbf{r}$ and multiplying Q_n to some vector $\mathbf{r}$. We first recall from §2 that the cost of solving $P_n\mathbf{y} = \mathbf{r}$ is of $O(n \log n) + O(\ell n)$ operations. To compute $Q_n\mathbf{r}$, we make use of the partitioning (8). Note that R_n in (8) is a matrix containing only $2s-1$ nonzero entries; we therefore need $O(s)$ operations for computing $R_n\mathbf{r}$. Since T_n is a Toeplitz matrix, $T_n\mathbf{r}$ can be computed in $O(n \log n)$ operations by embedding T_n into a $2n$-by-$2n$ circulant matrix, see Strang [23]. Hence $Q_n\mathbf{r}$ can be obtained in $O(n \log n)$ operations. Thus the number of operations required for each iteration of the PCGS method is of order $O(n \log n)$.

Finally, we consider the memory requirement. We note that in addition to some n-vectors, we have to store only the first column (or eigenvalues) of the matrices T_n, $\mathcal{T}_n[(z-1)^\ell]$ and $\mathcal{C}_n[b]$ but not the whole matrices. Thus we need $O(n)$ memory for the PCGS method.

4. Numerical results. In this section, we test the performance of our Toeplitz-circulant preconditioners P_n for solving Toeplitz systems and the queueing problems discussed in §3. All computations were done by Matlab on an HP 715 workstation.

For the tests on Toeplitz systems, we tried the following generating functions:

$$\text{(i)}\ g_1(z) = \frac{(z^4-1)}{(z-\frac{3}{2})(z-\frac{1}{2})} = \frac{15}{8}\sum_{k=1}^{\infty}\frac{1}{(2z)^k} + \frac{13}{24} + \frac{7}{36}z - \frac{11}{54}z^2 - \frac{65}{24}\sum_{k=3}^{\infty}\left(\frac{2z}{3}\right)^k.$$

$$\text{(ii)}\ g_2(z) = \frac{(z+1)^2(z-1)^2}{(z-\frac{3}{2})(z-\frac{1}{2})} = -\frac{9}{8}\sum_{k=1}^{\infty}\frac{1}{(2z)^k} + \frac{5}{24} + \frac{47}{36}z + \frac{29}{54}z^2 - \frac{25}{24}\sum_{k=3}^{\infty}\left(\frac{2z}{3}\right)^k.$$

$$\text{(iii)}\ g_3(z) = \frac{(z+1)^2(z-1)}{(z-\frac{3}{2})(z-\frac{1}{2})} = \frac{9}{4}\sum_{k=1}^{\infty}\frac{1}{(2z)^k} + \frac{11}{12} - \frac{7}{18}z - \frac{25}{12}\sum_{k=2}^{\infty}\left(\frac{2z}{3}\right)^k.$$

Clearly the functions $g_i, i = 1, 2, 3$, all have zeros on $|z| = 1$. We note that the preconditioners proposed in Chan [6] and Chan and Tang [9] are not applicable here because g_i are complex-valued functions on $|z| = 1$.

We note that the Toeplitz matrices formed by g_i's are nonsymmetric, therefore the systems $\mathcal{T}_n[g_i]\mathbf{x} = \mathbf{b}$ are solved by the PCGS method; see Sonneveld [22]. The stopping criterion we used is

$$(16) \qquad \frac{||\mathbf{r}_k||_2}{||\mathbf{r}_0||_2} < 10^{-6},$$

where $\mathbf{r}_k$ is the residual at the kth iteration. The right-hand side vector is $(1, 1, \dots, 1)^t$ and the initial guess is the zero vector. Table 1 gives the numbers of iterations required for convergence by using preconditioners I, P_n, and $\mathcal{C}_n[g]$. The symbol ** there denotes that the method does not converge in 5000 iterations. We see that the circulant preconditioner does not work well when the generating function has zeros on $|z| = 1$ and that the number of iterations required for convergence actually grows with n. However, our preconditioner P_n gives very fast convergence in all cases and the rate is actually improving with increasing n.

Next we test our preconditioner for queueing networks mentioned in §3. Since Q_n in (7) is irreducibly diagonally dominant, both the Jacobi and Gauss-Seidel methods converge

TABLE 1
Numbers of iterations for different preconditioners.

	g_1			g_2			g_3		
n	I	P_n	$C_n[g_1]$	I	P_n	$C_n[g_2]$	I	P_n	$C_n[g_3]$
8	8	7	8	8	8	7	8	9	7
16	22	6	9	26	7	9	26	5	12
32	68	5	9	132	6	11	68	6	12
64	315	4	9	**	6	14	202	5	13
128	3417	4	10	**	5	15	573	5	17
256	**	4	10	**	5	18	**	5	22
512	**	4	10	**	5	25	**	5	28

TABLE 2
Numbers of iterations for $\lambda_j = 1/2^j$.

s	1				4				$n-1$			
n	I	P_n	$C_n[g]$	J	I	P_n	$C_n[g]$	J	I	P_n	$C_n[g]$	J
8	8	5	6	95	8	5	7	78	8	6	6	45
16	15	4	6	115	15	5	7	86	13	7	8	70
32	28	4	7	213	27	5	8	209	21	7	9	114
64	**	4	7	307	**	5	8	306	48	7	10	173
128	**	3	7	470	**	5	8	469	**	7	10	267
256	**	3	8	768	**	5	8	768	**	7	10	434
512	**	3	8	1331	**	5	8	1331	**	6	10	746

when applied to solving the system (6). However, by using the partitioning of Q_n as in (8) and taking advantage of the Toeplitz matrix-vector multiplication (see §3.2), we see that each iteration of the Jacobi method can be done in $O(n \log n)$ operations, the same count as that of the PCGS method. This special property is not shared by the Gauss-Seidel method, which will still require $O(n^2)$ operations per iteration. Thus in our comparisons, we used only the Jacobi method.

We tried two sets of queueing parameters:

(i) $\lambda_j = \frac{1}{2^j}$, $j = 1, 2, \ldots$, and

(ii) $\lambda_j = \frac{90}{(\pi j)^4}$, $j = 1, 2, \ldots$.

We note that, in both cases, $\lambda = \sum_{k=1}^{\infty} \lambda_k = 1$. The service rate μ is set to $\mu = \lambda/s$. By (15), we see that $b(1) \neq 0$ and hence $\ell = 1$. Clearly, in both cases the mean arrival batch size $E(B)$ is finite because $\sum_j j\lambda_j < \infty$. Therefore, $b(z) \in \mathcal{W}$ and is given by (14). We remark that by using (13), the radius of convergence ρ for the first set of queueing parameters is 2. Hence regardless of the values of μ, its $b(z)$ will always be in $\mathcal{W}$.

The initial guess for both methods is $(1, 1, \ldots, 1)/n$. The stopping criteria for the PCGS method is again given by (16), whereas for the Jacobi method, it is $||\mathbf{x}_k - \mathbf{x}_{k-1}||_2 < 10^{-6}$, where $\mathbf{x}_k$ is the solution obtained at the kth iteration. Tables 2-3 give the numbers of iterations required for convergence for $s = 1, 4$ and $n - 1$. The symbol J means the Jacobi method is used. Again $**$ signifies that the method does not converge in 5000 iterations. In Table 4 the symbol "kflop" means 1000 floating point operations. Note that the case $s = n - 1$ is not covered by our Theorem 3.1. However, we note that in all the cases we tested, our preconditioner P_n is clearly the best choice.

5. Concluding remarks. We note that although we concentrate on the T. Chan circulant preconditioners here, the convergence results in Theorems 2.2 and 3.1 can easily be extended to include other circulant preconditioners. For instance, results for Strang's circulant preconditioners can be obtained if we replace Lemma 2.1 by theorems in [24]. In particular, using

TABLE 3
Numbers of iterations for $\lambda_j = 90/(\pi j)^4$.

s	1				4				$n-1$			
n	I	P_n	$C_n[g]$	J	I	P_n	$C_n[g]$	J	I	P_n	$C_n[g]$	J
8	8	5	6	389	8	5	7	264	8	6	8	112
16	16	4	7	1050	16	6	9	899	16	8	12	212
32	32	4	9	2253	32	6	10	2139	32	12	15	372
64	64	4	11	3431	64	5	12	3398	79	15	23	577
128	125	4	13	3874	124	5	15	3842	**	18	29	1005
256	365	4	17	**	388	5	18	**	**	21	34	1841
512	**	3	21	**	**	5	21	**	**	17	38	3163

TABLE 4
Numbers of kflops for $\lambda_j = 90/(\pi j)^4$ *and* $s = 1$.

n	I	P_n	$C_n[g]$	J
8	21	20	23	413
16	82	37	56	2410
32	340	80	150	11213
64	1437	172	388	37042
128	5997	373	977	90536
256	37423	807	2696	**
512	**	1421	7060	**

Theorem 6 from [24], we can show that if the quotient function $h(z)$ is a rational function of type (μ, ν), then our method converges in at most $(1 + 2\max\{\mu, \nu\} + \ell)$ steps for large n.

REFERENCES

[1] G. AMMAR AND W. GRAGG, *Superfast solution of real positive definite Toeplitz systems*, SIAM J. Matrix Anal. Appl., 9 (1988), pp. 61-76.

[2] O. AXELSSON AND V. BARKER, *Finite Element Solution of Boundary Value Problems: Theory and Computation*, Academic Press, New York, 1984.

[3] F. DI BENEDETTO, *Analysis of preconditioning techniques for ill-conditioned Toeplitz matrices*, SIAM J. Sci. Comput., 16 (1995), pp. 682-697.

[4] F. DI BENEDETTO, G. FIORENTINO, AND S. SERRA, *C.G. Preconditioning for Toeplitz matrices*, Comput. Math. Appl., 25 (1993), pp. 35-45.

[5] J. BUNCH, *Stability of methods for solving Toeplitz systems of equations*, SIAM J. Sci. Statist. Comput., 6 (1985), pp. 349-364.

[6] R. CHAN, *Toeplitz preconditioners for Toeplitz systems with nonnegative generating functions*, IMA J. Numer. Anal., 11 (1991), pp. 333-345.

[7] R. CHAN AND T. CHAN, *Circulant preconditioners for elliptic problems*, J. Numer. Linear Algebra Appl., 1 (1992), pp. 77-101.

[8] R. CHAN, J. NAGY, AND R. PLEMMONS, *FFT-based preconditioners for Toeplitz-block least squares problems*, SIAM J. Numer. Anal., 30 (1993), pp. 1740-1768.

[9] R. CHAN AND P. TANG, *Fast band-Toeplitz preconditioners for Hermitian Toeplitz systems*, SIAM J. Sci. Comput., 15 (1994), pp. 164-171.

[10] R. CHAN AND M. YEUNG, *Circulant preconditioners for complex Toeplitz matrices*, SIAM J. Numer. Anal., 30 (1993), pp. 1193-1207.

[11] T. CHAN, *An optimal circulant preconditioner for Toeplitz systems*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 767-771.

[12] J. CONWAY, *Functions of one complex variable*, Springer-Verlag, Berlin, 1973.

[13] R. COOPER, *Introduction to Queueing Theory*, 2nd ed., Arnold, London, 1981.

[14] U. GRENANDER AND G. SZEGÖ, *Toeplitz Forms and Their Applications*, 2nd ed., Chelsea, New York, 1984.

[15] T. HUCKLE, *Circulant and skew-circulant matrices for solving Toeplitz matrix problems*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 767-777.

[16] T. Ku and C. Kuo, *Design and analysis of Toeplitz preconditioners*, IEEE Trans. Signal Process., 40 (1991), pp. 129-141.

[17] ———, *A minimum-phase LU factorization preconditioner for Toeplitz matrices*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 1470-1480.

[18] ———, *Spectral properties of preconditioned rational Toeplitz matrices: The nonsymmetric case*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 521-544.

[19] T. Oda, *Moment analysis for traffic associated with Markovian queueing systems*, IEEE Trans. Comm., 39 (1991), pp. 737-745.

[20] D. O'Leary and J. Simmons, *A bidiagonalization-regularization procedure for large scale discretizations of ill-posed problems*, SIAM J. Sci. Statist. Comput., 2 (1981), pp. 474-488.

[21] A. Seila, *Multivariate estimation of conditional performance measure in regenerative simulation*, Amer. J. Math. Mgt. Sci., 10 (1990), pp. 17-45.

[22] P. Sonneveld, *A fast Lanczos-type solver for non-symmetric linear systems*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 36-52.

[23] G. Strang, *A proposal for Toeplitz matrix calculations*, Stud. Appl. Math., 74 (1986), pp. 171-176.

[24] L. Trefethen, *Approximation theory and numerical linear algebra*, in Algorithms for Approximation II, J. Mason and M. Cox, eds., Chapman and Hall, London, 1990.

[25] W. Trench, *An algorithm for the inversion of finite Toeplitz matrices*, SIAM J. Appl. Math., 12 (1964), pp. 515-522.

[26] E. Tyrtyshnikov, *Optimal and super-optimal circulant preconditioners*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 459-473.

[27] R. Varga, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1963.

[28] J. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965.

015

TIMELY COMMUNICATION

Under the "timely communications" policy for the SIAM Journal on Scientific Computing, papers that have significant timely content and do not exceed five pages automatically will be considered for a separate section of the journal with an accelerated reviewing process. It will be possible for the note to appear approximately six months after the date of acceptance.

GREENGARD'S N-BODY ALGORITHM IS NOT ORDER N*

SRINIVAS ALURU†

Abstract. Greengard's N-body algorithm claims to compute the pairwise interactions in a system of N particles in $O(N)$ time for a fixed precision. In this paper, we show that the choice of precision is not independent of N and has a lower bound of $\log N$. We use this result to show that Greengard's algorithm is not $O(N)$.

Key words. Greengard's algorithm, N-body problem, particle simulation, tree algorithms for particle simulation

AMS subject classifications. 70-08, 70F15

1. Introduction.

The N-body problem constitutes simulating the motion of N particles under the influence of mutual gravitational interactions. Since the problem cannot be solved in closed form, a discrete approximation has to be used. The force on every particle due to the rest of the system is computed and this information is used to update the system over a small interval of time δt. This constitutes one iteration in the solution of an N-body problem. A straightforward computation of the interactions takes $O(N^2)$ time per iteration, which is prohibitive since physicists want to simulate the motion of large collections of particles over long time scales. A number of algorithms have been designed to reduce the time complexity per iteration. The main principle behind these algorithms is approximating the force between two collections of particles that are far apart, without having to compute every pairwise interaction. With the notable exception of Greengard [2], most researchers paid little attention to a rigorous worst-case analysis of the complexity of their algorithms. Greengard's algorithm remains the only algorithm so far with a proven worst-case complexity of $O(N)$.

We limit the discussion to the two-dimensional version of Greengard's algorithm for convenience and simplicity. The first step in Greengard's algorithm is creating a hierarchical subdivision of the space containing the particles constructed as follows: let D be the length of a square box enclosing all the particles. (Greengard assumes the box has length 1, which can easily be achieved by a change of units.) The box is subdivided into four boxes with a side length half of the original box. Boxes that do not contain any particles are discarded. Boxes containing a single particle (or k particles for some fixed $k > 0$) are left as they are. The rest of the boxes are recursively subdivided. This recursive subdivision is naturally represented by a tree. The tree has N leaves and at most four children per node.

Let s be the smallest interparticle distance. (The terminology used here is different from Greengard's.) Each subdivision of a box reduces the box length by half. To separate two particles that are s apart, we may need a box of length less than $\frac{s}{\sqrt{2}}$. The worst-case number of subdivisions required to separate two particles s apart in two dimensions is given by the smallest k for which $\frac{D}{2^k} < \frac{s}{\sqrt{2}}$. Consequently, the height of the tree is bounded by $O(\log \frac{D}{s})$ and the total number of nodes in the tree is bounded by $O(N \log \frac{D}{s})$. In three dimensions, the worst-case

*Received by the editors December 14, 1993; accepted for publication (in revised form) October 20, 1995.

†School of Computer and Information Science, Syracuse University, Syracuse, NY 13244-4100 (aluru@top.cis.syr.edu). This work is supported by the Applied Mathematical Sciences Program of the Ames Laboratory-U.S. Department of Energy under contract W-7405-ENG-82.

number of subdivisions required is the smallest k for which $\frac{D}{2^k} < \frac{s}{\sqrt{3}}$. Therefore, the bounds on the height of the tree and the number of nodes is the same as in the case of two dimensions.

Greengard's arguments can be summarized as follows: for a fixed machine precision, only certain classes of particle distributions can be modeled, independent of the algorithm used. Therefore, by restricting attention to only those particle distributions that can be modeled on a given machine, D can be bound by the largest floating point number representable on the machine and s has to be no less than the smallest floating point number representable. Thus, $\log \frac{D}{s}$ is bounded by the log of the ratio of the largest to the smallest floating point number representable, a constant termed p. The size of the tree is bounded by $O(pN)$. Greengard determines the running time of his algorithm to be $N(\alpha p^2 + \beta p + \gamma)$ in two dimensions and $N(\alpha p^4 + \beta p^2 + \gamma)$ in three dimensions, where α, β, and γ are constants. Since p is taken to be a constant, the algorithm is claimed to run in $O(N)$ time in two or three dimensions.

The above arguments imply that the height of the tree is bounded by $O(p)$, a constant. Yet we know that the height of a tree with N leaves and at most a constant number of children per node is $\Omega(\log N)$. How can this disparity be explained?

To further highlight the discrepancy, consider the first step in Greengard's algorithm—the construction of the tree representing the hierarchical subdivision. At every level of the tree, the nodes containing more than one particle (or more than a fixed number of particles) are subdivided and particles in each parent box are distributed among its child boxes. Since each particle is assigned to a box at every level and there are at most p levels, the work involved is proportional to Np. Since p is taken to be a constant, the complexity is computed to be $O(N)$.

Consider running this algorithm on a distribution such that each child box contains exactly a fourth of the particles of the parent box. The resulting tree is a quadtree with $\log N$ levels and the work involved in constructing the tree is easily seen to be $O(N \log N)$, not $O(N)$.

The problem lies in the assumption that the parameters D and s are entirely dependent on the spatial distribution of the particles and not related to the number of particles N. To understand why this assumption is invalid, consider the behavior of $\frac{D}{s}$ for a fixed N. In particular, we shall investigate the upper and lower bounds for $\frac{D}{s}$ as a function of N.

For any $N \geq 3$ particles, $\frac{D}{s}$ can be made arbitrarily large by reducing the distance between the closest particles (thus reducing s) or by increasing the spread of the particles (thus increasing D). This validates the argument that for a fixed machine precision, only certain classes of particle distributions can be modeled, independent of the algorithm used.

To minimize the ratio $\frac{D}{s}$ for a fixed N, all the particles should be at a distance of s from their nearest neighbors. To see why, suppose this is not true. We can reduce D by "moving in" particles that are farther than s from each other, while keeping s the same. Or else, we can increase s by increasing the distance between particles that are s apart, while keeping D unchanged. In either case, $\frac{D}{s}$ decreases contradicting minimality. Furthermore, the particles must be packed as closely as possible. Figure 1 shows the configuration minimizing the ratio $\frac{D}{s}$ for a fixed N in two dimensions. Each particle has six nearest neighbors in the close packing, all at a distance of s from it. The particle is at the center of the hexagon formed by its nearest neighbors. The particles fit exactly in a cell of size $D \times D$ (i.e., boundaries of the cell are inhabited by particles). Adding the particles columnwise,

$$N = \left\lfloor \frac{D}{s} + 1 \right\rfloor + \left\lfloor \frac{D}{s} \right\rfloor + \left\lfloor \frac{D}{s} + 1 \right\rfloor + \cdots \quad \left(\left\lfloor \frac{2D}{\sqrt{3}s} + 1 \right\rfloor \text{terms} \right),$$

$$N \leq \frac{D}{s}\left[\frac{2D}{\sqrt{3}s} + 1\right] + \frac{D}{\sqrt{3}s} + 1,$$

$$N \leq \frac{2}{\sqrt{3}}\frac{D^2}{s^2} + \left(1 + \frac{1}{\sqrt{3}}\right)\frac{D}{s} + 1,$$

$$\frac{D}{s} \geq c_1 N^{\frac{1}{2}}$$

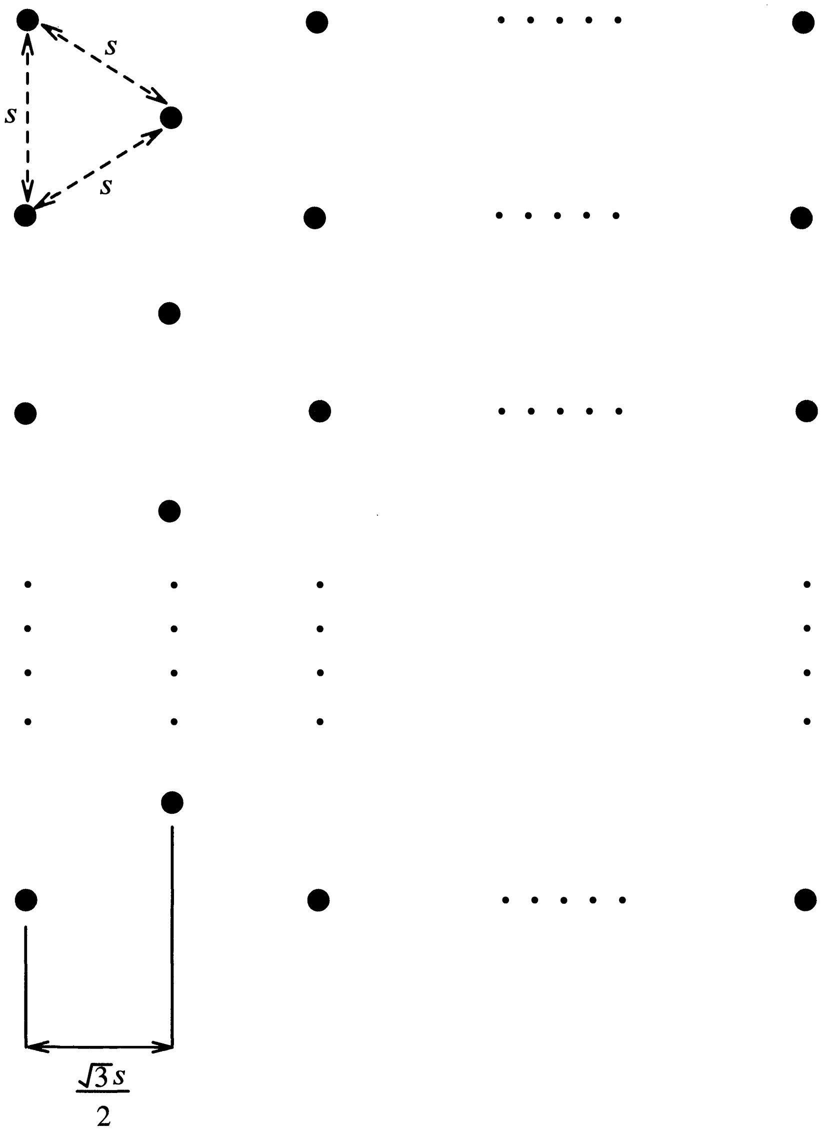


FIG. 1. *The configuration minimizing the ratio of the cell length D containing all the particles and the smallest interparticle distance s in two dimensions. The ratio $\frac{D}{s}$ is minimized when all particles are a distance s apart from their nearest neighbors and the particles are packed as closely as possible.*

for some constant c_1. Since this is computed using the configuration minimizing $\frac{D}{s}$, $\log \frac{D}{s}$ is $\Omega(\log N)$. Since $\log \frac{D}{s}$ is bounded by p, p is also $\Omega(\log N)$. In three dimensions, it can be shown that

$$\frac{D}{s} \geq c_2 N^{\frac{1}{3}}.$$

In this case also, $\log \frac{D}{s}$ is $\Omega(\log N)$ and p is $\Omega(\log N)$.

How does this translate to what classes of particle distributions can be modeled on a machine with precision parameter p? It is already noted that not all distributions can be modeled for any given $N \geq 3$ because of precision limits. However, unless $p \geq c \log N$ (c is a constant), no distribution can be modeled for that N. The very fact that we are able to run an N-body problem for a collection of N particles with precision parameter p implies that $p \geq c \log N$. Thus, p cannot be taken as a constant in the analysis of the running time of the algorithm, and Greengard's algorithm is not $O(N)$. Greengard's time complexity in two dimensions is $N(\alpha p^2 + \beta p + \gamma)$, which is $\Omega(N \log^2 N)$. In three dimensions, the complexity is $N(\alpha p^4 + \beta p^2 + \gamma)$, which is $\Omega(N \log^4 N)$. The running time matches the lower bound only for a uniform distribution. For arbitrary distributions, the running time is unbounded.

Acknowledgments. I wish to thank John Gustafson and G. M. Prabhu for discussion and helpful comments.

REFERENCES

[1] S. Aluru, *Distribution-Independent Hierarchical N-Body Methods*, Ph.D. thesis, Iowa State University, Ames, IA, 1994.

[2] L. Greengard, *The Rapid Evaluation of Potential Fields in Particles Systems*, MIT Press, Cambridge, MA, 1988.

TIMELY COMMUNICATION

Under the "timely communications" policy for the SIAM Journal on Scientific Computing, papers that have significant timely content and do not exceed five pages automatically will be considered for a separate section of the journal with an accelerated reviewing process. It will be possible for the note to appear approximately six months after the date of acceptance.

ON THE REMOVAL OF BOUNDARY ERRORS CAUSED BY RUNGE-KUTTA INTEGRATION OF NONLINEAR PARTIAL DIFFERENTIAL EQUATIONS*

SAUL ABARBANEL[†], DAVID GOTTLIEB[‡], AND MARK H. CARPENTER[§]

Abstract. The temporal integration of hyperbolic partial differential equations (PDEs) has been shown to lead sometimes to the deterioration of accuracy of the solution because of boundary conditions. A procedure for removal of this error in the linear case has been established previously.

In this paper we consider hyperbolic PDEs (linear and nonlinear) whose boundary treatment is accomplished via the simultaneous approximation term (SAT) procedure. A methodology is presented for recovery of the full order of accuracy and has been applied to the case of a fourth-order explicit finite-difference scheme.

Key words. Runge-Kutta scheme, temporal accuracy, time-dependent boundary conditions

AMS subject classifications. 65L06, 65M15, 65M20

1. Introduction. A growing interest is evident in long-time integration for solving problems in areas such as aeroacoustics, electromagnetics, and material science. The use of long-time integration [1] necessitates working with higher-order (fourth-order accuracy and above) schemes. Often the methodology of choice is to semidiscretize the equations by applying a high-order (fourth-order and above) spatial difference operator and then advancing temporally with single-level multistage Runge-Kutta (RK) integrators. The question arises of how to supply boundary values at the intermediate stages of the RK integrations. For hyperbolic PDEs, time-dependent conditions must be imposed at the inflow boundary.

The conventional (and intuitive) method of imposing inflow boundary conditions (BCs) at the intermediate stages is to use the "appropriate" value of the boundary data $g(t)$ at each stage. Thus, for example, at a stage that corresponds to $t + \frac{\Delta t}{2}$ one would impose $g(t + \frac{\Delta t}{2})$.

In a previous paper [2], when applied to hyperbolic PDEs with time-dependent BCs, the procedure described above was shown to reduce the accuracy near the inflow boundary to first order; thus, the overall accuracy could not exceed $O(\Delta x)^2$. This conclusion was independent of the order of accuracy of the spatial difference operator.

One way to avoid the dilemma of determining which boundary values to supply at the intermediate stages is to advance the RK integration without imposing any intermediate values and obtain the intermediate boundary values from the numerical solution operator. However, this approach reduces substantially the stability limit (e.g., the allowable time step is reduced

*Received by the editors March 6, 1995; accepted for publication (in revised form) November 9, 1995. The research of the first two authors was supported by the National Aerodynamic and Space Administration under NASA contract NAS1-19480 while the authors were in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA 23681. The second author was supported also by AFOSR 95-1-0074, ARPA grant N00014-91-J-4016, NSF grants DMS-9500814 and DE-FG-02-95ER25239.

[†] Department of Mathematical Sciences, Division of Applied Mathematics, Tel-Aviv University, Tel-Aviv, Israel (saul@math.tau.ac.il).

[‡] Division of Applied Mathematics, Brown University, Providence, RI 02912 (dig@cfm.brown.edu).

[§] Aeronautic and Acoustic Methods Branch, Fluid Mechanics and Acoustics Division, NASA Langley Research Center, Hampton, VA 23681-0001 (carpentr@cmb21.larc.nasa.gov).

by a factor of 2 in the case of a fourth-order classic RK scheme with a fourth-order spatial derivative operator); hence, this method is not desirable.

In [2], a general methodology was presented in the case of linear PDEs for the correct imposition of the intermediate-stage boundary values to enable the scheme to recover its full formal accuracy. This method was expounded in detail for the case of the classic fourth-order RK integration with a hyperbolic fourth-order spatial difference operator. In [2] it was shown that in the nonlinear case (e.g., hyperbolic conservation laws) this methodology was applicable to RK integration up to third order. For RK methods of fourth order and above, we were unable to extend the theoretical approach described in [2].

In this paper, we address anew the issue of dealing with the nonlinear case. We present a methodology for retaining the full accuracy even in the nonlinear case. The application of this methodology involves numerical determination of free parameters in contradistinction to the linear procedure described in [2]. We find, for example, that for the fourth-order classic RK integrator with fourth-order explicit spatial derivative operator, the full accuracy is retained without any reduction in the allowable time step.

The new procedure is demonstrated for hyperbolic problems in which the BCs are satisfied with the SAT approach [4]. We consider the SAT procedure because it is the only one that prevents temporal growth not present in the true solution of a system of PDEs. Section 2 describes how to correctly apply the intermediate SAT BCs in the case of a linear problem. In §3 we address the nonlinear case.

2. The linear case. In this section we analyze the effect of imposing inflow boundary conditions by conventional methods for cases in which the discretization algorithm employs the SAT approach. (See [3].) The SAT is a penalty type of method that was constructed to ensure that the numerical solution does not include temporal growth that is not of a physical origin. This construction is achieved by mimicking the energy estimate of the PDE.

We consider the following hyperbolic problem [3]:

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0, \quad 0 \leq x \leq t, \quad t \geq 0, \tag{2.1}$$

$$u(0,t) = g(t). \tag{2.2}$$

The SAT formulation for the semidiscrete version of (2.1) and (2.2), based on a uniform grid, is

$$\frac{dv_i(t)}{dt} = \left[\frac{1}{h} D \vec{V}(t)\right]_i - \frac{1}{h}\tau \vec{\sigma}\, [v_0(t) - g(t)], \quad i = 0, 1, \ldots, N, \quad t \geq 0, \tag{2.3}$$

where $\vec{V} = [v_0, v_1, \ldots, v_N]^T$ is the semidiscrete approximation that converges to $u(x_i, t)$ at the spatial grid points x_i (for stable discretizations) and $\frac{1}{h}D$ is the differential matrix representation of the derivative operator $-\frac{\partial}{\partial x}$. The vector $\vec{\tau\sigma}$ depends on the differentiation matrix $\frac{1}{h}D$, and on the energy norm used in bounding the error. This vector is determined as described in [3]; see the discussion after (6).

The demonstration of accuracy deterioration will be shown for the four-stage "classic" RK algorithm, which is one of the most commonly used RK time-advancement schemes. So that the analysis makes sense, we assume that the spatial discretization is at least fourth-order accurate.

The above-mentioned four-stage RK integrator is implemented as follows ($\lambda = \frac{\Delta t}{\Delta x}$):

$$V^{(1)} = V^{(n)} + \frac{\lambda}{2} D V^{(n)} - \frac{\lambda}{2}\tau \vec{\sigma}\, [v_0^{(n)} - g(t)], \tag{2.4}$$

$$V^{(2)} = V^{(n)} + \frac{\lambda}{2} D V^{(1)} - \frac{\lambda}{2} \tau \vec{\sigma} \left[v_0^{(1)} - g\left(t + \frac{\Delta t}{2}\right)\right], \tag{2.5}$$

$$V^{(3)} = V^{(n)} + \lambda D V^{(2)} - \lambda \tau \vec{\sigma} \left[v_0^{(2)} - g\left(t + \frac{\Delta t}{2}\right)\right], \tag{2.6}$$

$$\begin{aligned} V^{(n+1)} &= V^{(n)} + \frac{\lambda}{6} D [V^{(n)} + 2V^{(1)} + 2V^{(2)} + V^{(3)}] \\ &\quad - \frac{\lambda}{6} \tau \vec{\sigma} \left\{ [v_0^{(n)} - g(t)] + 2\left[v_0^{(1)} - g\left(t + \frac{\Delta t}{2}\right)\right] + 2\left[v_0^{(2)} - g\left(t + \frac{\Delta t}{2}\right)\right] \right. \\ &\quad \left. + [v_0^{(3)} - g(t + \Delta t)] \right\}. \end{aligned} \tag{2.7}$$

To check for accuracy we substitute for $V^{(n)}$ the exact values $u(x_i, t)$ and, in particular, $v_0^{(n)} = g(t)$. From (2.4), one can see that on the boundary (using the differential equations (2.1) and (2.2)), we have

$$v_0^{(1)} = v_0^{(n)} - \frac{\Delta t}{2} \left(\frac{\partial}{\partial x} u \right)_0 = g(t) + \frac{\Delta t}{2} g'(t). \tag{2.8}$$

Thus, in (2.7) we have, for the term $v_0^{(1)} - g\left(t + \frac{\Delta t}{2}\right)$,

$$v_0^{(1)} - g\left(t + \frac{\Delta t}{2}\right) = \left[g(t) + \frac{\Delta t}{2} g'(t) \right] - g\left(t + \frac{\Delta t}{2}\right) = 0(\Delta t)^2. \tag{2.9}$$

Thus, $V^{(n+1)} - V^{(n)}$ is at best $O(\Delta t)^2$, rather than $O(\Delta t)^5$, which was expected from the RK scheme used.

In this linear case, the remedy proposed in the previous paper [2] works here as well. In particular, (2.4) - (2.7) take the following form:

$$V^{(1)} = V^{(n)} + \frac{\lambda}{2} D V^{(n)} - \frac{\lambda}{2} \tau \vec{\sigma} \, [v_0^{(n)}(t) - g(t)], \tag{2.10}$$

$$V^{(2)} = V^{(n)} + \frac{\lambda}{2} D V^{(1)} - \frac{\lambda}{2} \tau \vec{\sigma} \left[v_0^{(1)} - g(t) - \frac{\Delta t}{2} g'(t) \right], \tag{2.11}$$

$$V^{(3)} = V^{(n)} + \lambda D V^{(2)} - \lambda \tau \vec{\sigma} \left[v_0^{(2)} - g(t) - \frac{\Delta t}{2} g'(t) - \frac{\Delta t^2}{4} g''(t) \right], \tag{2.12}$$

$$\begin{aligned} V^{n+1} &= V^n + \frac{\lambda}{6} D \left[V^{(n)} + 2V^{(1)} + 2V^{(2)} + V^{(3)} \right] \\ &\quad - \frac{\lambda}{6} \tau \vec{\sigma} \left\{ [v_0^{(n)} - g(t)] + 2\left[v_0^{(1)} - g(t) - \frac{\Delta t}{2} g'(t) \right] \right. \\ &\quad + 2\left[v_0^{(2)} - g(t) - \frac{\Delta t}{2} g'(t) - \frac{\Delta t^2}{4} g''(t) \right] \\ &\quad \left. + \left[v_0^{(3)} - g(t) - \Delta t g' - \frac{\Delta t^2}{2} g'' - \frac{\Delta t^3}{4} g''' \right] \right\}. \end{aligned} \tag{2.13}$$

We can readily verify that $V^{(n+1)} - V^{(n)} = 0(\Delta t)^5$, as required.

3. The nonlinear case. For simplicity, first we consider the scalar conservation law PDE:

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0, \qquad 0 \le x \le 1, \quad t \ge 0, \tag{3.1}$$

$$u(0, t) = g(t). \tag{3.2}$$

In general, for any spatial discretization (whether explicit or implicit), the semidiscrete form of (3.1) and (3.2) is

$$\frac{dV}{dt} = \frac{1}{h} Df(V) - \frac{1}{h}\tau \vec{\sigma} \, [v_0 - g(t)]. \tag{3.3}$$

Using the notation of [3],

$$D = -P^{-1}Q, \tag{3.4}$$

where $\frac{1}{h}D$ is the differentiation matrix that represents the differential operator $(-\frac{\partial}{\partial x})$; this matrix is composed of the explicit part Q and the inverse of the implicit part P. For a fully explicit spatial differentiation, $P = I + B$, where B differs from zero only at the two diagonal corners. (See examples of P and Q in [4].)

The vector $\vec{\sigma}$, again with the notation in [4], is given by

$$\vec{\sigma} = ha(u_0) g_{00} P^{-1} H^{-1} S,$$

where $a(u_0) = (\frac{\partial f}{\partial u})_{x=0}$ and g_{00} is twice the value of the left upper corner element of HQ. For the definition of the matrix H, see assumption I in [1]. The parameter τ is determined from the stability consideration to be $\tau \ge 1$. (See [3].)

Next, we demonstrate that writing the classic fourth-order RK integration for (3.3), using the linear "fix" as in (2.10) - (2.13), does not yield the required fourth-order accuracy:

$$V^{(1)} = V^{(n)} + \frac{\lambda}{2} Df(V^{(n)}) - \frac{\lambda}{2}\tau \vec{\sigma} \, [v_0^{(n)} - g(t)], \tag{3.5}$$

$$V^{(2)} = V^{(n)} + \frac{\lambda}{2} Df(V^{(1)}) - \frac{\lambda}{2}\tau \vec{\sigma} \left[v_0^{(1)} - g(t) - \frac{\Delta t}{2} g'(t) \right], \tag{3.6}$$

$$V^{(3)} = V^{(n)} + \lambda Df(V^{(2)}) - \lambda\tau \vec{\sigma} \left[v_0^{(2)} - g(t) - \frac{\Delta t}{2} g'(t) - \frac{\Delta t^2}{4} g''(t) \right], \tag{3.7}$$

$$\begin{aligned} V^{(n+1)} = {} & V^{(n)} + \frac{\lambda}{6} D[f(V^n) + 2f(V^{(1)}) + 2f(V^{(2)}) + f(V^{(3)})] \\ & -\frac{\lambda}{6}\tau \vec{\sigma} \left\{ [v_0^{(n)} - g(t)] + 2\left[v_0^{(1)} - g(t) - \frac{\Delta t}{2} g'(t) \right] \right. \\ & +2\left[v_0^{(2)} - g(t) - \frac{\Delta t}{2} g'(t) - \frac{\Delta t^2}{4} g''(t) \right] \\ & \left. + \left[v_0^{(3)} - g(t) - \Delta t g' - \frac{\Delta t^2}{2} g'' - \frac{\Delta t^3}{4} g''' \right] \right\}. \end{aligned} \tag{3.8}$$

Again, in checking the accuracy, we take $V^{(n)} = u(x_i, t)$ and, in particular, $v_0^{(n)} = g(t)$; furthermore, $\frac{1}{h}Df(V^n) = -\frac{\partial}{\partial x} f(u) + O(\Delta t)^4$. With these preliminary calculations, we immediately obtain the following from (3.5):

$$v_0^{(1)} = v_0^n - \frac{\Delta t}{2}\frac{\partial}{\partial x} f(\mu)_0 = g(t) + \frac{\Delta t}{2}\frac{\partial}{\partial t}(u)_0 = g(t) + \frac{\Delta t}{2} g'(t). \tag{3.9}$$

Note that this result is the same for the linear case. (See (2.8).) Thus, with $v_0^{(n)} = g(t)$ and (3.9), we can supply, for the purpose of checking accuracy, the correct values of $v^{(n)}$ and $v_0^{(1)}$. When we examine (3.6) using the above results, the governing nonlinear PDE, and a simple Taylors expansion, we have

$$\begin{aligned}
V^{(2)} &= V^{(n)} + \frac{\lambda}{2} Df\left[V^{(n)} + \frac{\lambda}{2} Df(V^n)\right] \\
&= V^{(n)} - \frac{\Delta t}{2}\frac{\partial}{\partial x} f\left[u - \frac{\Delta t}{2}\frac{\partial}{\partial x} f(u) + O(\Delta t)^5\right] \\
&= V^n - \frac{\Delta t}{2}\frac{\partial}{\partial x} f\left[u + \frac{\Delta t}{2}\frac{\partial u}{\partial t} + O(\Delta t)^5\right] \\
&= V^{(n)} - \frac{\Delta t}{2}\frac{\partial}{\partial x}\left\{f(u) + \frac{\Delta t}{2}\frac{\partial f}{\partial u}\frac{\partial u}{\partial t} + \frac{\Delta t^2}{8}\left(\frac{\partial^2 f}{\partial u^2}\right)\left(\frac{\partial u}{\partial t}\right)^2 + O(\Delta t)^3\right\} \\
&= V^{(n)} - \frac{\Delta t}{2}\left\{\frac{\partial f}{\partial x} + \frac{\Delta t}{2}\frac{\partial}{\partial t}\left(\frac{\partial f}{\partial x}\right)\right\} + O(\Delta t)^3 \\
&= u - \frac{\Delta t}{2}\left[-u_t - \frac{\Delta t}{2} u_{tt}\right] + O(\Delta t)^3. \qquad (3.10)
\end{aligned}$$

Finally, on the boundary we have

$$v_0^{(2)} = g(t) + \frac{\Delta t}{2} g'(t) + \frac{\Delta t^2}{4} g''(t) + 0(\Delta t)^3. \qquad (3.11)$$

From (3.11) we see that the "penalty" term in (3.8) introduces an error of $(\Delta t)^3$. Because the coefficient of $(\Delta t)^3$, $[f_{uu}(u_t)^2]_0$, cannot generally be expressed as a function of $g(t)$ and its derivatives, this situation is difficult to remedy. Thus, the "linear procedure" fails at the third RK stage.

We now propose a methodology for dealing with the RK integration of nonlinear hyperbolic conservation laws. We first present this procedure in the case of the classic fourth-order RK scheme. Our starting point is the observation that the "linear procedure" yields the required accuracy for $v_0^{(n)}$ and $v_0^{(1)}$. At each stage, the idea is to use a linear combination of the "linear" SAT or penalty terms used in (3.5) and (3.6). Thus, the fourth-order classic RK stages will be

$$V^{(1)} = V^{(n)} + \frac{\lambda}{2} Df(V^n) - \frac{\lambda}{2}\alpha\, \vec{\sigma}\, [v_0^{(n)} - g(t)], \qquad (3.12)$$

$$V^{(2)} = V^{(n)} + \frac{\lambda}{2} Df(V^{(1)}) - \frac{\lambda}{2}\beta\, \vec{\sigma}\left[v_0^{(1)} - g(t) - \frac{\Delta t}{2} g'(t)\right] - \frac{\lambda}{2}\gamma\, \vec{\sigma}\, [v_0^{(n)} - g(t)], \qquad (3.13)$$

$$V^{(3)} = V^{(n)} + \lambda Df(V^{(2)}) - \lambda\delta\, \vec{\sigma}\, [v_0^{(n)} - g(t)] - \lambda\varepsilon\, \vec{\sigma}\left[v_0^{(1)} - g(t) - \frac{\Delta t}{2} g'(t)\right], \qquad (3.14)$$

$$\begin{aligned}
V^{(n+1)} &= V^{(n)} + \frac{\lambda}{6} D[f(V^n) + 2f(V^{(1)}) + 2f(V^{(2)}) + f(V^{(3)})] \\
&\quad - \frac{\lambda}{6}\mu\, \vec{\sigma}\, [v_0^{(n)} - g(t)] - \frac{\lambda}{3}\nu\, \vec{\sigma}\left[v_0^{(1)} - g(t) - \frac{\Delta t}{2} g'(t)\right], \qquad (3.15)
\end{aligned}$$

where the free parameters α, β, γ, δ, ε, μ, and ν will be chosen to maximize the allowable time step. The previous discussion clearly shows that the system ((3.12) - (3.15)) maintains the fourth-order accuracy. The question remains of whether the CFL stability condition deteriorates in comparison with the conventional application of the RK integration. Also, optimal choice of the free parameters $\alpha, \ldots, \nu$ clearly varies with the spatial discretizations (i.e., the differentiation matrix D) and boundary closures. The absolute values of the eigenvalues of the amplification matrix that result from (3.12) - (3.15) should not exceed unity. We carried out this procedure (with Mathematica software) in the case of an explicit fourth-order algorithm with third-order boundary closure (in V). In this case, $H = I$, and the matrices P and Q can be found in §9.1 of [4]. The CFL condition becomes $\lambda \leq 2.1$ with the following values of the free parameters: $\alpha = \beta = -\varepsilon = 1, \delta = 2, \mu = 0, \nu = 3$, and $\gamma = -0.37$. This restriction on the time step Δt is the same as that for the linear problem with "conventional" (i.e., less accurate) boundary values with or without the SAT term.

4. Conclusions. In summary, we have a fourth-order RK scheme in (3.12) - (3.15) applied to a nonlinear PDE that maintains overall fourth-order accuracy without a decrease in the allowable time step. The extension to a system of hyperbolic PDEs is quite straightforward with the use of the SAT approach delineated in [3].

REFERENCES

[1] O. Kreiss and J. Oliger, *Comparison of accurate methods for the integration of hyperbolic problems*, Tellus, 24 (1972), pp. 199-215.

[2] M. H. Carpenter, D. Gottlieb, S. Abarbanel, and W. S. Don, *The theoretical accuracy of Runge–Kutta time discretizations for the initial boundary value problem: A study of the boundary error*, SIAM J. Sci. Comput., 16 (1995), pp. 1241-1252.

[3] M. H. Carpenter, D. Gottlieb, and S. Abarbanel, *Time-stable boundary conditions for finite-difference schemes solving hyperbolic systems: Methodology and application to high-order compact schemes*, JCP, 111 (1994), pp. 220-236.

[4] ———, *Stable and accurate boundary treatments for compact, high order finite difference schemes*, Appl. Numer. Math., 12 (1993), pp. 55-87.

COMPARISONS OF LATTICE BOLTZMANN AND FINITE DIFFERENCE METHODS FOR A TWO-DIMENSIONAL VISCOUS BURGERS EQUATION*

BRACY H. ELTON[†]

Abstract. Lattice Boltzmann methods have been proposed as a computational means for solving various partial differential equations. We look at three lattice Boltzmann methods and an analogous finite difference method for solving a two-dimensional scalar, viscous Burgers equation with periodic boundary conditions. First, using monotonicity arguments we prove that in the ℓ_1 norm the lattice Boltzmann methods converge first-order temporally and second-order spatially. Then we provide some computational results which substantiate the theoretical results. Finally, we compare the lattice Boltzmann and finite difference methods. The basis of comparison includes numerical results with regard to accuracy, order of convergence, performance, and timing measurements; memory requirements; and conservations laws.

Key words. consistency, convection–diffusion, convergence, explicit, finite difference, Hilbert expansion, lattice Boltzmann, monotone, nonlinear, parallel, periodic, stability, viscous Burgers equation

AMS subject classifications. 35F25, 65M06, 65M12, 65Y20, 76R99, 82C40

1. Introduction.

Lattice gas and lattice Boltzmann methods have been proposed as computational means, efficient on parallel and vector architectures, for solving a variety of partial differential equations [3]. We will look at three lattice Boltzmann methods for solving the two-dimensional (2-D) nonlinear convection–diffusion equation

$$(1) \qquad \frac{\partial}{\partial t}\rho + \rho\frac{\partial}{\partial x}\rho = \nu\left(\frac{\partial^2}{\partial x^2}\rho + \frac{\partial^2}{\partial y^2}\rho\right),$$

which is a two-dimensional scalar, viscous Burgers equation, for $\rho = \rho(t, \mathrm{x})$, $\mathrm{x} = (x, y) \in [0, L]^2$, diffusion coefficient ν, initial condition $\rho(0, \mathrm{x}) = \rho_I(\mathrm{x})$, and periodic boundary conditions. We introduce the methods and establish theoretically their convergence, verify computationally the theoretical results, and compare the lattice Boltzmann methods and an analogous finite difference method.

The underlying purpose of this paper is to present a fairly thorough analysis of some simple lattice Boltzmann methods and some computational techniques that can be used to study more complex methods in the absence of strong theoretical results. It is hoped that those investigating more complex lattice Boltzmann and lattice gas methods, for example, ones for the Navier–Stokes equations and those with more complex boundary conditions, can and will employ some of the techniques described herein to evaluate (at least numerically) their models and methods.

The paper is outlined as follows. Section 2 introduces the lattice Boltzmann methods and establishes the theory of their convergence. Under certain conditions we show that in the ℓ_1 norm the methods are spatially second-order, temporally first-order, conservative, monotone difference methods for computing solutions to (1). The theoretical results are obtained by applying the key results of [7]. Section 3 contains detailed numerical studies and comparisons of the lattice Boltzmann methods with each other and with an analogous finite difference method. The numerical studies include such aspects as domains of monotonicity, accuracy, order of convergence, performance and timing measurements, and memory requirements; also included are conservation laws and collision rules and their effect on accuracy. The numerical studies also substantiate the theoretical results. The performance and timing measurements

*Received by the editors December 9, 1993; accepted for publication (in revised form) February 21, 1995. Copyright ©1995 B. H. Elton.

[†]SuperComputer Group, Fujitsu America, Inc., 3055 Orchard Drive, San Jose, CA 95134–2022 (bracy.elton@na-net.ornl.gov).

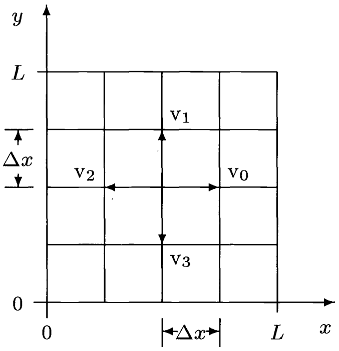


FIG. 1. *A 2-D lattice.*

were taken on Fujitsu VPX240 and VPP500 systems. Finally, concluding remarks are made in §4.

2. Theory. We will be discussing three lattice Boltzmann methods, enumerated LB1, LB2, and LB3, for computing solutions to (1) with periodic boundaries. LB1 and LB2 are introduced here; LB3 was introduced in earlier work [4, 5].

Although our lattice Boltzmann methods could be motivated by analogous lattice gas methods, we describe them directly. In our analysis, we will be applying (and borrowing) much of the notation, properties, and theorems of [7], in which key theorems and properties of convective–diffusive lattice Boltzmann methods appear. Note that for purposes of increasing readability, specific details of the sometimes rather involved analyses have been deferred to the appendices.

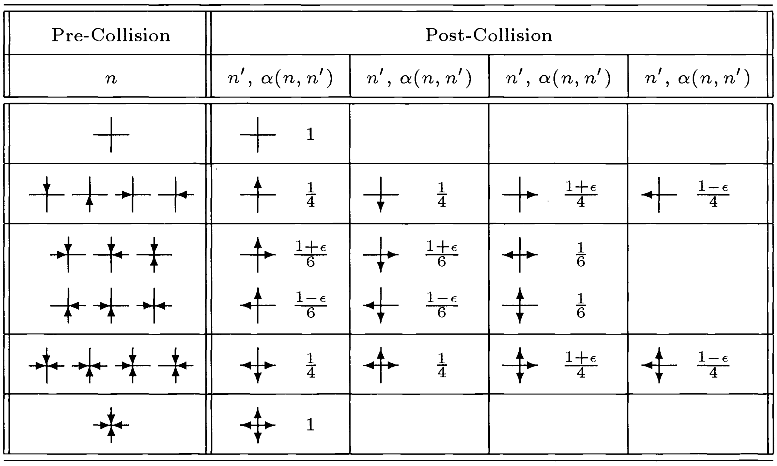
2.1. The lattice Boltzmann methods. The following are the essential elements of our 2-D lattice Boltzmann methods, each of which abides by an exclusion principle that restricts the number of particles per velocity per node.[1]

1. A 2-D doubly periodic (toroidal) regular lattice, $\mathbf{X} = [0, L]^2$, where L is the spatial scale length (Figure 1).
2. The set of unit velocity vectors $\mathbf{V}=\{\mathrm{v}_k \mid k \in \{0,1,2,3\}\} \subset \mathbf{R}^2$, where
$$\mathrm{v}_0 = (+1, 0)\,, \quad \mathrm{v}_1 = (0, +1)\,, \quad \mathrm{v}_2 = (-1, 0)\,, \quad \mathrm{v}_3 = (0, -1)\,.$$
3. $|\mathbf{V}| = 4$ links per node, each link having length Δx, where $|\mathbf{V}|$ is the cardinality of $\mathbf{V}$.
4. Mean occupation numbers: for each time step, lattice node, and velocity, there is an associated mean occupation number, $F \in [0, 1]^{\mathbf{V}} \equiv \mathcal{I}$.[2] The mean occupation number can be viewed as a normalized particle distribution, i.e., the likelihood of the presence of a particle, as in the Boltzmann equation, e.g., [1].
5. An advection operator, $\mathcal{A}$, that advances particle distributions to neighboring nodes and discrete time steps of duration $\Delta t > 0$, where $\Delta t = (\Delta x)^2/(4\nu)$ with ν the diffusion coefficient of (1). (The specification of Δt is an artifact of the construction of the lattice Boltzmann methods via a discrete Hilbert expansion, as explained in §2.3.)
6. A collision operator, $\mathcal{C}(F)$, that determines the nature of interactions between particle distributions at nodes of the lattice.

[1] We use bold typefaces to indicate a set, e.g., $\mathbf{R} = \{reals\}$.

[2] $[0, 1]^{\mathbf{V}}$ is the set of functions from $\mathbf{V}$ to the closed interval $[0, 1]$.

TABLE 1
*Collision rules for LB*1.



Pre-Collision	Post-Collision			
n	n', $\alpha(n,n')$	n', $\alpha(n,n')$	n', $\alpha(n,n')$	n', $\alpha(n,n')$
	1			
	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1+\epsilon}{4}$	$\frac{1-\epsilon}{4}$
	$\frac{1+\epsilon}{6}$ $\frac{1-\epsilon}{6}$	$\frac{1+\epsilon}{6}$ $\frac{1-\epsilon}{6}$	$\frac{1}{6}$ $\frac{1}{6}$	
	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1+\epsilon}{4}$	$\frac{1-\epsilon}{4}$
	1			

The mean occupation number F of a lattice node denotes the expected number of particles with a particular velocity at that node. Let $F(m, \mathbf{i}, \mathrm{v}) \in [0, 1]$ denote the expected number of particles with velocity v at time step m and lattice node $\mathbf{i} \in \mathbf{X}$. Thus, $F(m, \mathbf{i}, \mathrm{v})$ is in the single particle phase space $\mathbf{X} \times \mathbf{V}$ at time step m ; it evolves according to a kinetic equation of the form, $\mathcal{A}F = F + \mathcal{C}(F)$, where $\mathcal{A}$ models the movement of particle distributions (advection operator) and $\mathcal{C}(F)$, operating only on v, models the interactions of particle distributions (collision operator).

The collision operator for such lattice Boltzmann methods is written according to the formula $\mathcal{C}(F) \equiv \sum_{n,n'\in \mathbf{2}^{\mathbf{V}}} \alpha(n,n')(n'-n)F^n\bar{F}^{\bar{n}}$, where $\mathbf{2} \equiv \{0, 1\}$, and F^n and $\bar{F}^{\bar{n}}$ are defined by $F^n \equiv \prod_{\mathrm{v}\in\mathbf{V}}(F(\mathrm{v}))^{n(\mathrm{v})}$ and $\bar{F}^{\bar{n}} \equiv \prod_{\mathrm{v}\in\mathbf{V}}(1 - F(\mathrm{v}))^{(1-n(\mathrm{v}))}$, an overbarred quantity meaning one minus that quantity.[3] Requirements on $\alpha(n, n')$ are that $\alpha(n, n') \in [0, 1]$ and that it maintains conservation of probability, i.e., $\sum_{n'\in\mathbf{2}^{\mathbf{V}}} \alpha(n, n') = 1$, for all $n \in \mathbf{2}^{\mathbf{V}}$.

The advection operator $\mathcal{A}$ is defined by $\mathcal{A}F(m, \mathbf{i}, \mathrm{v}) = F(m + 1, \mathbf{i} + \mathrm{v}, \mathrm{v})$. Finally, the behavior of a lattice Boltzmann method is characterized by the microdynamical evolution equation

$$(2) \qquad \mathcal{A}F(m, \mathbf{i}, \mathrm{v}) = F(m, \mathbf{i}, \mathrm{v}) + \mathcal{C}(F(m, \mathbf{i}, \cdot)(F(m, \mathbf{i}, \mathrm{v}))\,,$$

which states that the new mean occupation numbers (on the left) at the new locations $(m+1, \mathbf{i}+\mathrm{v})$ are the same as the mean occupation numbers at the current location $(m, \mathbf{i})$ plus collisional corrections. This is analogous to a certain type of differencing of the Boltzmann equation, i.e., forward Euler (first order in time), central conservative finite difference in space (second order in space), and fixed in velocity (velocities are restricted to the directions available on the particular lattice and a fixed set of predetermined speeds, which in our case is $\{\Delta x/\Delta t\}$). Note that $F \mapsto F + \mathcal{C}(F) \in [0, 1]^{\mathbf{V}}$.

Collision rules for LB1, LB2, and LB3 appear in Tables 1, 3, and 4. Therein the rules are given pictorially. As an example, collision rules for LB1 are listed formally in Table 2.

Our lattice Boltzmann methods are then defined as the finite difference methods that, given the collision rules specified in Tables 1–4, evolve according to (2), in which $F(0, \mathbf{i}, \mathrm{v})$

[3] $\mathbf{2}^{\mathbf{V}}$ is the set of functions from $\mathbf{V}$ to $\mathbf{2}$.

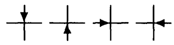
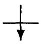
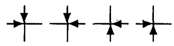
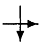
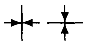
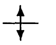
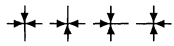
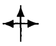
TABLE 2

*Formal listing of collision rules for LB*1. *For each input state n in each type of collision, the table lists the corresponding possible output states n' with nonzero probabilities $\alpha(n, n')$, where $a = \frac{1+\epsilon}{2}$ and $\bar{a} = \frac{1-\epsilon}{2}$.*

Collision	n				n'				$\alpha(n, n')$
Type	$n(\mathrm{v}_0)$	$n(\mathrm{v}_1)$	$n(\mathrm{v}_2)$	$n(\mathrm{v}_3)$	$n'(\mathrm{v}_0)$	$n'(\mathrm{v}_1)$	$n'(\mathrm{v}_2)$	$n'(\mathrm{v}_3)$	
No Particles	0	0	0	0	0	0	0	0	1
1 Particle	0	0	0	1	0	0	0	1	$1/4$
	0	0	1	0	0	0	1	0	$\bar{a}/2$
	0	1	0	0	0	1	0	0	$1/4$
	1	0	0	0	1	0	0	0	$a/2$
2 Particles	0	0	1	1	0	0	1	1	$\bar{a}/3$
	0	1	1	0	0	1	1	0	$\bar{a}/3$
	1	1	0	0	1	1	0	0	$a/3$
	1	0	0	1	1	0	0	1	$a/3$
	1	0	1	0	1	0	1	0	$1/6$
	0	1	0	1	0	1	0	1	$1/6$
3 Particles	1	1	1	0	1	1	1	0	$1/4$
	1	1	0	1	1	1	0	1	$a/2$
	1	0	1	1	1	0	1	1	$1/4$
	0	1	1	1	0	1	1	1	$\bar{a}/2$
4 Particles	1	1	1	1	1	1	1	1	1

TABLE 3

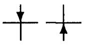
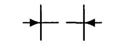
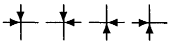
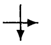
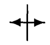
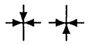
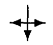
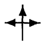
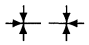
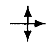
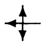
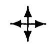
*Collision rules for LB*2.

Pre-Collision	Post-Collision			
n	n', $\alpha(n, n')$	n', $\alpha(n, n')$	n', $\alpha(n, n')$	n', $\alpha(n, n')$
	1			
	$\frac{1}{4}$	 $\frac{1}{4}$	$\frac{1+\epsilon}{4}$	$\frac{1-\epsilon}{4}$
	$\frac{1+\epsilon}{4}$	 $\frac{1+\epsilon}{4}$	$\frac{1-\epsilon}{4}$	$\frac{1-\epsilon}{4}$
	$\frac{1}{2}$	 $\frac{1}{2}$		
	$\frac{1}{4}$	 $\frac{1}{4}$	$\frac{1+\epsilon}{4}$	$\frac{1-\epsilon}{4}$
	1			

$= F_I(\mathbf{i}, \mathrm{v})$, for all $\mathbf{i} \in \mathbf{X}$ and for all $\mathrm{v} \in \mathbf{V}$, where F_I is the initial condition. Approximate solutions to (1) are obtained through a special combination of the mean occupation numbers $F(m, \mathbf{i}, \mathrm{v})$, i.e., the average over the set of velocities, as is determined by the local conservation law.

2.2. Conservation, equilibria, and the H-theorem. Note that the lattice Boltzmann methods LB1, LB2, and LB3, which are governed by (2), each possess the local conservation law of conservation of mass, the locally conserved quantity being $\sum_{\mathrm{v}\in\mathbf{V}} F(m, \mathbf{i}, \mathrm{v})$. Also, because of the assumed periodic nature of the lattice, i.e., periodic boundary conditions, the presence of a local conservation law gives rise to a global conservation law. The globally conserved quantity is the sum of the mean occupation numbers over the entire lattice, i.e., $\sum_{\mathbf{i}\in\mathbf{X}} \sum_{\mathrm{v}\in\mathbf{V}} F(m, \mathbf{i}, \mathrm{v})$.

TABLE 4
Collision rules for LB3. Each line in the table lists possible input states n and corresponding output states n′ and their nonzero probabilities $\alpha(n, n')$.

Pre-Collision	Post-Collision			
n	n', $\alpha(n,n')$	n', $\alpha(n,n')$	n', $\alpha(n,n')$	n', $\alpha(n,n')$
	1			
	$\frac{1}{2}$	$\frac{1}{2}$		
	$\frac{1+\epsilon}{2}$	$\frac{1-\epsilon}{2}$		
	$\frac{1+\epsilon}{4}$	 $\frac{1+\epsilon}{4}$	$\frac{1-\epsilon}{4}$	$\frac{1-\epsilon}{4}$
	1			
	 1			
	 $\frac{1}{2}$	 $\frac{1}{2}$		
	 $\frac{1+\epsilon}{2}$	 $\frac{1-\epsilon}{2}$		
	 1			

Note that using $\epsilon = K\delta$, for constant K, a convection scaling parameter the collision operators for LB1, LB2, and LB3 can be written in the form, $\mathcal{C}(F) = \mathcal{C}^{(0)}(F) + \delta\mathcal{C}^{(1)}(F)$, where $\delta > 0$ is a dimensionless parameter and $\mathcal{C}^{(0)}(F)$ and $\mathcal{C}^{(1)}(F)$ are generalized Boltzmann operators in terms of $\mathcal{S}^{(0)}(n, n')$ and $\mathcal{S}^{(1)}(n, n')$, each not depending on δ, where $\mathcal{S}(n, n') = \mathcal{S}^{(0)}(n, n') + \delta\mathcal{S}^{(1)}(n, n')$ and $\mathcal{C}^{(0)}(F)$ is in semidetailed balance, i.e., $\sum_{n' \in 2^{\mathrm{V}}} \mathcal{S}^{(0)}(n', n) = \sum_{n' \in 2^{\mathrm{V}}} \mathcal{S}^{(0)}(n, n') = 1$, for every $n \in \mathbf{2}^{\mathrm{V}}$.

The H-theorem for diffusive lattice Boltzmann methods [7, Thm. 2.2] states in part that at an equilibrium, $\mathcal{C}^{(0)}(F) = 0$. For LB1, LB2, and LB3, we find that at equilibrium $F(\mathrm{v}_0) = F(\mathrm{v}_1) = F(\mathrm{v}_2) = F(\mathrm{v}_3) \equiv u$.

2.3. Hilbert expansion and diffusion. For the purposes of conducting our analyses, let us review some of the main points regarding the Hilbert expansion and diffusion from [7]. Let $\delta > 0$ be a small parameter and assume the lattice spacing $\Delta x = \delta L$ and the time step $\Delta t = \delta^2 T$ for some $L, T > 0$. The Taylor expansion of $\mathcal{A}H(t, \mathrm{x}, \mathrm{v})$ about $(t, \mathrm{x}, \mathrm{v})$ is $\mathcal{A}H(t, \mathrm{x}, \mathrm{v}) = H(t + \Delta t, \mathrm{x} + \mathrm{v}\Delta x, \mathrm{v}) = \sum_{j=0}^{\infty} \frac{1}{j!}\left[\delta^2 T \partial_t + \delta L \mathrm{v} \cdot \nabla\right]^j H(t, \mathrm{x}, \mathrm{v})$. Note that the terms can be grouped by order of δ.

A solution H of the lattice Boltzmann method (2) can be regarded as a function of the variables t, x, and v. We postulate that H can be expanded in the form

$$H(t, \mathrm{x}, \mathrm{v}) = \sum_{k=0}^{\infty} \delta^k h^{(k)}(t, \mathrm{x}, \mathrm{v}), \tag{3}$$

where $h^{(0)}$ is an equilibrium solution of $\mathcal{C}^{(0)}(H)$. The series (3) is called a *Hilbert expansion* of H. We examine conditions for the existence of a Hilbert expansion of H under the assumption that H is infinitely differentiable.

Define

$$a(\delta) = \mathcal{A}H - H = (\mathcal{A} - I)\sum_{k=0}^{\infty} \delta^k h^{(k)},$$

$$c(\delta) = \mathcal{C}(H) = \mathcal{C}\left(\sum_{k=0}^{\infty} \delta^k h^{(k)}\right) = \mathcal{C}^{(0)}\left(\sum_{k=0}^{\infty} \delta^k h^{(k)}\right) + \delta\mathcal{C}^{(1)}\left(\sum_{k=0}^{\infty} \delta^k h^{(k)}\right).$$

Expanding $a = a(\delta)$ and $c = c(\delta)$ about $\delta = 0$ in Taylor series gives

$$a(\delta) = a^{(0)} + \delta\, a^{(1)} + \delta^2 a^{(2)} + \delta^3 a^{(3)} + \cdots, \tag{4}$$

$$c(\delta) = c^{(0)} + \delta\, c^{(1)} + \delta^2 c^{(2)} + \delta^3 c^{(3)} + \cdots. \tag{5}$$

Matching the terms order by order yields

$$H(t+\Delta t, \mathrm{x}+\mathrm{v}\Delta x, \mathrm{v}) - H(t, \mathrm{x}, \mathrm{v}) = \mathcal{C}(H) = \mathcal{C}^{(0)}(h^{(0)}) + \sum_{j=1}^{\infty}(g^{(j)} - \mathcal{L}\cdot h^{(j)})\delta^j \equiv \sum_{j=0}^{\infty} \mathcal{T}_j \delta^j, \tag{6}$$

where $\mathcal{L} = \mathcal{D}\mathcal{C}^{(0)}(h^{(0)})$ is the so-called *linearized collision operator* and the $g^{(j)}$ are the remaining terms.

Since H is purported to be a solution of the lattice Boltzmann equation (2), it follows that $\mathcal{T}_j = 0$, or, equivalently,

$$\mathcal{C}(h^{(0)}) = 0, \tag{7}$$

$$\mathcal{L}\cdot h^{(j)} = g^{(j)}, \quad j = 1, 2, \ldots. \tag{8}$$

Thus, the asymptotic series (3) is constructed by solving each of the linear systems in (8).

For each of LB1, LB2, and LB3, the linearized collision operator $\mathcal{L}$ is the derivative, i.e., the Jacobian, of the $\mathcal{O}[1]$ portion of the collision operator, i.e., $\mathcal{C}^{(0)}(H)$, evaluated at equilibrium and to which the H-theorem is applied. In each method, $\mathcal{L}$ is singular. (Please see the appendices for the details.) Hence, the Hilbert series can be constructed if and only if $g^{(k)} \in \mathrm{Range}(\mathcal{L})$, in which case

$$h^{(j)} = \mathcal{L}^{+}\cdot g^{(j)} + s^{(j)}, \tag{9}$$

where $s^{(j)} \in \mathrm{Kernel}(\mathcal{L})$ is arbitrary and $\mathcal{L}^{+}$ is the pseudoinverse of $\mathcal{L}$. Note that each $\mathcal{L}$ is symmetric and nonpositive definite in the equilibrium solution u. Further, although the eigenvectors are the same, i.e., the eigenmatrix for each method is given by

$$\mathbf{Q} = [\mathbf{q}_0, \mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3] = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & -1 \\ 1 & -1 & 0 & 1 \\ 1 & 0 & -1 & -1 \end{bmatrix},$$

their respective eigenvalues differ. The eigenvalues are denoted $\boldsymbol{\lambda} = (\lambda_0, \lambda_1, \lambda_2, \lambda_3)$. It turns out for all three methods that $\lambda_0 = 0$ is the only zero eigenvalue; hence, $\mathcal{L}$ is indeed singular for each of the methods, and the pseudoinverse $\mathcal{L}^{+}$ of each method is given by $\mathcal{L}^{+} = \sum_{k=1}^{3} \frac{1}{\lambda_k} \frac{\mathbf{q}_k \mathbf{q}_k^T}{\mathbf{q}_k^T \mathbf{q}_k}$.

2.4. Consistency, stability, and convergence.

2.4.1. General definitions and theorems. For completeness, we include the definitions and statements of the theorems related to consistency, stability, and convergence. Interested readers can find the relevant proofs in [7], which also includes some theory for more general lattice Boltzmann methods.

Consider the finite sums $H^{(q)}(m, \mathbf{i}, \cdot) = \sum_{k=0}^{q} \delta^k h^{(k)}$, where $h^{(k)} = h^{(k)}(m, \mathbf{i}, \cdot)$ are grid functions.

Definition 2.1. *Let $q > 0$ be a fixed integer and $B_{\Delta t}$ a finite-dimensional Banach space with ℓ_1 norm $\|\cdot\|_{\Delta t}$.*

1. Consistency: *Let $\mathcal{A}H^{(q)} - H^{(q)} - \mathcal{C}[H^{(q)}] = \mathcal{T}^{(q)}$. If $\lim_{\Delta t \to 0} \frac{1}{\Delta t} \|\mathcal{T}^{(q)}(m, \cdot, \cdot)\|_{\Delta t} = 0$, then the lattice Boltzmann method is said to be consistent.*
2. Convergence: *If $F(m, \cdot, \cdot) \in B_{\Delta t}$ is the solution to the lattice Boltzmann method* (2) *and $\lim_{\Delta t \to 0} \|F(m, \cdot, \cdot) - H^{(q)}(m, \cdot, \cdot)\|_{\Delta t} = 0$, for all integers m such that $0 \leq m\Delta t \leq T$, then the lattice Boltzmann method is said to be convergent.*
3. Stability: *Define the block diagonal matrix $L_{\Delta t} : B_{\Delta t} \to B_{\Delta t}$ where the* **i***th diagonal block is defined by $\mathcal{L}(F, H^{(q)}) \cdot G = \int_0^1 \left[I + \mathcal{DC}\big((1-s)F + sH^{(q)}\big)\right] \cdot G\, ds$, for $G \in \mathcal{I} \equiv [0, 1]^{\mathbf{V}}$. The lattice Boltzmann method is said to be stable if for some $\tau > 0$, the family of matrices $\{\prod_{k=0}^{m} L_{\Delta t} : 0 < \Delta t < \tau$ and $0 \leq m\Delta t \leq T\}$, is uniformly bounded.*

Theorem 2.2. *Suppose a lattice Boltzmann method is consistent. Then stability is a sufficient condition for convergence.*

Using the ideas of monotone difference methods [12], sufficient conditions for stability of convective–diffusive lattice Boltzmann methods have been established [7]. The advection operator $\mathcal{A}$ can be interpreted as a vector-valued function defined on $\mathcal{I}$ having the vth coordinate function given by $\mathcal{A}[F](\mathrm{v}) = F(\mathrm{v}) + \mathcal{C}[F](\mathrm{v})$. The derivative of $\mathcal{A}$ is the $|\mathbf{V}| \times |\mathbf{V}|$ Jacobian matrix $\mathcal{J}_{\mathcal{A}}(F)$ whose v, wth element is

$$\frac{\partial}{\partial F(\mathrm{w})} \mathcal{A}[F](\mathrm{v}) . \tag{10}$$

Definition 2.3. *Let $\mathcal{E} = \prod_{k=0}^{|\mathbf{V}|} [M_-^{(k)}, M_+^{(k)}] \subseteq \mathcal{I}$ be a $|\mathbf{V}|$-dimensional interval upon which $\mathcal{J}_{\mathcal{A}}$ is nonnegative. That is, $F \in \mathcal{E}$ implies $\mathcal{J}_{\mathcal{A}}(F) \geq 0$. Then $\mathcal{E}$ is called a* domain of monotonicity *of the lattice Boltzmann method* (2)*, and the vectors $\mathbf{M}_- = \left[M_-(\mathrm{v})\right]_{\mathrm{v} \in \mathbf{V}}$ and $\mathbf{M}_+ = \left[M_+(\mathrm{v})\right]_{\mathrm{v} \in \mathbf{V}}$ are called the* extreme points *of $\mathcal{E}$.*

The following lemma demonstrates the invariance property of the advection operator on a domain of monotonicity.

Lemma 2.4. *Let $\mathcal{E}$ be a domain of monotonicity with extreme points $\mathbf{M}_-$ and $\mathbf{M}_+$ for a lattice Boltzmann method* (2)*. Suppose $\mathcal{C}[F](\mathbf{M}_-) = \mathcal{C}[F](\mathbf{M}_+) = 0$. Then $\mathcal{A}$ leaves $\mathcal{E}$ invariant. That is, $F \in \mathcal{E}$ implies $\mathcal{A}F \in \mathcal{E}$.*

The following is a stability condition.

Theorem 2.5. *Let $\mathcal{E}$ be a domain of monotonicity for a lattice Boltzmann method* (2)*. Let the extreme points $\mathbf{M}_-$ and $\mathbf{M}_+$ be such that $\mathcal{C}[F](\mathbf{M}_-) = \mathcal{C}[F](\mathbf{M}_+) = 0$. Suppose $F(0, \mathbf{i}, \cdot)$ and $H^{(q)}(0, \mathbf{i}, \cdot)$ are vectors in $\mathcal{E}$. If $\mathcal{C}(F)$ conserves mass locally, i.e., $F(\mathrm{v}) = \sum_{\mathrm{v} \in \mathbf{V}} \mathcal{C}(F)(\mathrm{v}) = 0$, then the method is stable.*

2.4.2. Convergence. From Theorems 2.2 and 2.5, we can obtain second-order (in δ) convergence of LB1 and LB2 in the ℓ_1 norm for sufficiently smooth initial conditions $F(0, \mathrm{x}, \cdot)$ that are wholly contained within each method's respective domain of monotonicity by (i) proving consistency and (ii) showing that the purported domain of monotonicity is indeed a domain of monotonicity. Similar arguments apply for LB3, except that Lemma 2.4 does not apply directly, making the arguments somewhat more involved. This case will be treated separately. The following standard notation is used: $\nabla \equiv (\partial_x, \partial_y)$ and $\overline{\nabla} \equiv (\partial_x, -\partial_y)$.

THEOREM 2.6 (Consistency of LB1). *Let* $\Delta x = L\delta$, $\nu = \frac{L^2}{4T}$, $\Delta t = T\delta^2$, $c = \frac{KL}{2T}$, $\epsilon = \frac{c\Delta x}{2\nu}$, *and* $A(u) = u(1-u)$. *Suppose*

$$\partial_t u + c\partial_x A(u) = \nu\nabla^2 u\,, \tag{11}$$

$$\partial_t \sigma^{(2)} + 6c\partial_x[A'(u)\sigma^{(2)}] = \nu\nabla^2\sigma^{(2)} - \mathcal{F}\,, \tag{12}$$

$$\sigma^{(1)} = \sigma^{(3)} = 0\,, \tag{13}$$

where $\mathcal{F} = \mathcal{F}(T, K, L, t, x, y, u, u_x, u_y, u_{xx}, u_{xy}, u_{yy}, u_{xxx}, u_{xyy}, u_{xxxx}, u_{xxyy}, u_{yyyy})$, *which is given in Appendix* A, *is a smooth function. Then* $g^{(j)} \in \mathrm{Range}(\mathcal{L})$, $j = 1, 2, 3, 4$, *and LB*1 *is consistent.*

Proof. Part (i): First we show $g^{(j)} \in \mathrm{Range}(\mathcal{L})$, for $j = 1, 2, 3, 4$. The first part of the proof follows by carrying out the expansions in (4), (5), (7), and (8) to get that $g^{(j)} = \sum_{k=1}^{3} c_k^{(j)}\mathbf{q}_k$. The specific values of $c_k^{(j)}$ are given in Appendix A. Applying (11)–(13) to the coefficients $c_k^{(j)}$ yields the desired result.

Part (ii): Now, we show consistency. Let $h^{(j)}$, $j = 0, 1, 2, 3, 4$, be defined by $h^{(0)} = (u, u, u, u)$ and $h^{(j)} = \mathcal{L}^+ \cdot g^{(j)} + \sigma^{(j)}$, $j = 1, 2, 3, 4$, as per (9). Consider the grid function $H^{(3)}(m, \mathbf{i}, \cdot) = \sum_{j=0}^{3} h^{(j)}(m, \mathbf{i}, \cdot)\,\delta^j$. Then $\mathcal{A}H^{(3)} - H^{(3)} - \mathcal{C}(H^{(3)}) = -\mathbf{T}(H^{(3)}) +$ higher-order terms, where $\mathbf{T}(H^{(3)}) = \sum_{j=0}^{4} \mathbf{T}^{(j)}\delta^j$.

Since $\mathbf{T}^{(0)} = \mathcal{L}\cdot h^{(0)}$ and $\mathbf{T}^{(j)} = \mathcal{L}\cdot h^{(j)} - g^{(j)}$, $j = 1, 2, 3$, it follows that $\mathbf{T}^{(j)} = 0$, $j = 0, 1, 2, 3$. Moreover, $\mathbf{T}^{(4)} = -\sum_{k=0}^{3} c_k^{(4)}\mathbf{q}_k = -c_3^{(4)}\mathbf{q}_3$, where $c_3^{(4)}$ is given in Appendix A. Consequently, if $u \in C^4([0,1]^2, (0,1))$ and $\sigma^{(2)} \in C^2(\mathbf{R}^2, \mathbf{R})$, then $\mathcal{T}(H^{(3)}) = \mathcal{O}[\delta^4]$ and the method will be consistent. Since u and $\sigma^{(2)}$ are solutions to the uniformly parabolic equations (11) and (12), respectively, it follows that they and their corresponding derivatives are uniformly bounded and, consequently, consistency is in fact true. □

THEOREM 2.7 (Consistency of LB2). *Let* $\Delta x = L\delta$, $\nu = \frac{L^2}{4T}$, $\Delta t = T\delta^2$, $c = \frac{KL}{2T}$, $\epsilon = \frac{c\Delta x}{2\nu}$, *and* $A(u) = u(1-u)$. *Suppose*

$$\partial_t u + c\partial_x A(u) = \nu\nabla^2 u\,, \tag{14}$$

$$\partial_t \sigma^{(2)} + 6c\partial_x(A'(u)\sigma^{(2)}) = \nu\nabla^2\sigma^{(2)} - \mathcal{F}\,, \tag{15}$$

$$\sigma^{(1)} = \sigma^{(3)} = 0\,, \tag{16}$$

where $\mathcal{F}$ *is a smooth function of* u *and its spatial derivatives, as defined in Appendix* B. *Then* $g^{(j)} \in \mathrm{Range}(\mathcal{L})$, $j = 1, 2, 3, 4$, *and LB*2 *is consistent.*

Proof. The proof is similar to that for Theorem 2.6, with the supporting details appearing in Appendix B. □

Showing consistency of LB3 is slightly different than of LB1 and LB2 due to $\sigma^{(3)} = 0$ not being a solution to the method's $\mathcal{O}[\delta^5]$ consistency condition (see (20) below and Appendix C).

THEOREM 2.8 (Consistency of LB3). *Let* $\Delta x = L\delta$, $\nu = \frac{L^2}{4T}$, $\Delta t = T\delta^2$, $c = \frac{KL}{2T}$, $\epsilon = \frac{c\Delta x}{2\nu}$, $A(u) = u(1-u)$, *and* $\lambda = -4A(u)$. *Suppose*

$$\partial_t u + c\partial_x A(u) = \nu\nabla^2 u\,, \tag{17}$$

$$\sigma^{(1)} = 0\,, \tag{18}$$

$$\partial_t \sigma^{(2)} + 6c\partial_x(A'(u)\sigma^{(2)}) = \nu\nabla^2\sigma^{(2)} - \mathcal{F}\,, \tag{19}$$

$$\partial_t \sigma^{(3)} + \tfrac{c}{2}\partial_x[A'(u)\sigma^{(3)}] = \nu\nabla^2\sigma^{(3)} - \mathcal{G}\,, \tag{20}$$

where for $u \in C^4((0,1)^2,(0,1))$, $\mathcal{F}$ *and* $\mathcal{G}$ *are smooth functions of* (a) T, K, L, *and* u *and its derivatives, and* (b) T, K, L, *and* u *and* $\sigma^{(2)}$ *and their derivatives, respectively, as defined in Appendix* C. *Then* $g^{(j)} \in \mathrm{Range}(\mathcal{L})$, $j = 1,2,3,4$, *and LB*3 *is consistent.*

Proof. Part (i): First we show $g^{(j)} \in \mathrm{Range}(\mathcal{L})$, for $j = 1,2,3,4$. The first part of the proof follows by carrying out the expansions in (4), (5), (7), and (8) to get that $g^{(j)} = \sum_{k=1}^{3} c_k^{(j)} \mathbf{q}_k$. The specific values of $c_k^{(j)}$ are given in Appendix C. Applying (17)–(20) to the coefficients $c_k^{(j)}$ yields the desired result.

Part (ii): Now, we show consistency. Let $h^{(j)}$, $j = 0,1,2,3,4$, be defined by $h^{(0)} = (u,u,u,u)$ and $h^{(j)} = \mathcal{L}^{+} \cdot g^{(j)} + \sigma^{(j)}$, $j = 1,2,3,4$, as per (9). Consider the grid function $H^{(3)}(m,\mathbf{i},\cdot) = \sum_{j=0}^{3} h^{(j)}(m,\mathbf{i},\cdot)\,\delta^j$. Then $\mathcal{A}H^{(3)} - H^{(3)} - \mathcal{C}(H^{(3)}) = -\mathbf{T}(H^{(3)}) +$ (higher-order terms), where $\mathbf{T}(H^{(3)}) = \sum_{j=0}^{4} \mathbf{T}^{(j)}\delta^j$.

Since $\mathbf{T}^{(0)} = \mathcal{L} \cdot h^{(0)}$ and $\mathbf{T}^{(j)} = \mathcal{L} \cdot h^{(j)} - g^{(j)}$, $j = 1,2,3$, it follows that $\mathbf{T}^{(j)} = 0$, $j = 0,1,2,3$. Moreover, $\mathbf{T}^{(4)} = -\sum_{k=0}^{3} c_k^{(4)} \mathbf{q}_k$, where the $c_k^{(4)}$ are given in Appendix C. Consequently, if $u \in C^4((0,1)^2,(0,1))$, $\sigma^{(2)} \in C^2(\mathbf{R}^2,\mathbf{R})$, and $\sigma^{(3)} \in C^2(\mathbf{R}^2,\mathbf{R})$, then $\mathcal{T}(H^{(3)}) = \mathcal{O}[\delta^4]$, and the method will be consistent. Since u, $\sigma^{(2)}$, and $\sigma^{(3)}$ are solutions to the uniformly parabolic equations (17), (19), and (20), they and their corresponding derivatives are uniformly bounded and consistency is in fact true. □

THEOREM 2.9 (Domain of monotonicity for LB1). *The set* $\mathcal{E} = [M_-, M_+]^{|\mathbf{V}|}$, *where* $M_- = 0$ *and* $M_+ = 1$, *is a domain of monotonicity for LB*1. *Let* $\mathbf{M}_- = M_-^{\mathbf{V}}$ *and* $\mathbf{M}_+ = M_+^{\mathbf{V}}$. *Then* $\mathcal{C}[F](\mathbf{M}_-) = \mathcal{C}[F](\mathbf{M}_+) = 0$.

Proof. Part (i): First we show that $\mathcal{E}$ is a domain of monotonicity. Let $f(x,y,z) = \frac{3+\epsilon[3-2(x+y+z)]}{12}$ and $g(x,y,z) = \frac{3-\epsilon[3+2(x+y+z)]}{12}$. Let $F_k = F(\mathrm{v}_k), k = 0,1,2,3$. We find that the Jacobian $\mathcal{J}_{\mathcal{A}}(F)$, which is defined generally by (10), in the case of LB1 is given by

$$\mathcal{J}_{\mathcal{A}}(F) = \begin{bmatrix} f(F_1,F_2,F_3) & f(F_0,F_2,F_3) & f(F_0,F_1,F_3) & f(F_0,F_1,F_2) \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ g(F_1,F_2,F_3) & g(F_0,F_2,F_3) & g(F_0,F_1,F_3) & g(F_0,F_1,F_2) \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{bmatrix}.$$

Consequently, showing that $\mathcal{J}_{\mathcal{A}}(F) \geq 0$ for $F \in \mathcal{E}$ is equivalent to showing that each of the two functions $f(x,y,z)$ and $g(x,y,z)$ is nonnegative for $M_- = 0 \leq x,y,z \leq M_+ = 1$. We now show that the local and boundary minima of f and g are nonnegative. If we examine the gradients of these functions, then we see that they are nonzero. Hence, there are no interior minima.

Standard variational arguments on the boundary easily establish that

$$\min \tfrac{1}{12}\{3(1+\epsilon), 3(1-\epsilon), 3+\epsilon, 3-\epsilon\} \leq f, g \leq \max \tfrac{1}{12}\{3(1+\epsilon), 3(1-\epsilon), 3+\epsilon, 3-\epsilon\}.$$

Since $\epsilon \in [-1,1]$, it quickly follows that $0 \leq f, g \leq \frac{1}{2}$.

Part (ii): It follows from (22) (in Appendix A) that $\mathcal{C}[F](\mathbf{M}_+) = \mathcal{C}[F](\mathbf{M}_-) = 0$. □

THEOREM 2.10 (Domain of monotonicity for LB2). *The set* $\mathcal{E} = [M_-, M_+]^{|\mathbf{V}|}$, *where* $M_- = 0$ *and* $M_+ = 1$, *is a domain of monotonicity for LB*2. *Let* $\mathbf{M}_- = M_-^{\mathbf{V}}$ *and* $\mathbf{M}_+ = M_+^{\mathbf{V}}$. *Then* $\mathcal{C}[F](\mathbf{M}_-) = \mathcal{C}[F](\mathbf{M}_+) = 0$.

Proof. Part (i): First we show $\mathcal{E}$ is a domain of monotonicity. Let $f(x) = \frac{1+\epsilon}{4} - \frac{\epsilon}{2}x$ and $g(x) = \frac{1-\epsilon}{4} + \frac{\epsilon}{2}x$. Let $F_k = F(\mathrm{v}_k), k = 0,1,2,3$. We find that the Jacobian $\mathcal{J}_{\mathcal{A}}(F)$ for LB2 is given by

$$\mathcal{J}_{\mathcal{A}}(F) = \begin{bmatrix} f(F_2) & f(F_3) & f(F_0) & f(F_1) \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ g(F_2) & g(F_3) & g(F_0) & g(F_1) \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{bmatrix}.$$

Consequently, showing that $\mathcal{J}_{\mathcal{A}}(F) \geq 0$ for $F \in \mathcal{E}$ is equivalent to showing that each of the two functions $f(x)$ and $g(x)$ is nonnegative for $M_- = 0 \leq x, y, z \leq M_+ = 1$. We now show that the local and boundary minima of f and g are nonnegative. If we examine the gradients of these functions, then we see that they are nonzero. Hence, there are no interior minima.

Standard variational arguments on the boundary easily establish that

$$\min\{\tfrac{1+\epsilon}{4}, \tfrac{1-\epsilon}{4}\} \leq f, g \leq \max\{\tfrac{1+\epsilon}{4}, \tfrac{1-\epsilon}{4}\}.$$

Since $\epsilon \in [-1, 1]$, it quickly follows that $0 \leq f, g \leq \frac{1}{2}$.

Part (ii): It follows from (23) (in Appendix B) that $\mathcal{C}[F](\mathbf{M}_+) = \mathcal{C}[F](\mathbf{M}_-) = 0$. □

THEOREM 2.11 (Domain of monotonicity for LB3). *The set* $\mathcal{E} = [M_-, M_+]^{|\mathrm{V}|}$, *where* $M_- = 0$ *and* $M_+ = \frac{1}{2}$, *is a domain of monotonicity for LB3.*

Proof. Let $f_1(x, y, z) = \frac{1+\epsilon}{2}(1 - 2y) + xy - xz + yz$, $f_2(x, y, z) = \frac{1-\epsilon}{2}(1 - 2y) + xy - xz + yz$, $g(x, y, z) = x - xy - xz + yz$ and $h(x, y, z) = \frac{1}{2} - y + xy - xz + yz$. Let $F_k = F(\mathrm{v}_k)$, $k = 0, 1, 2, 3$. The Jacobian $\mathcal{J}_{\mathcal{A}}(F)$ for LB3 is given by

$$\mathcal{J}_{\mathcal{A}}(F) = \begin{bmatrix} f_1(F_1, F_2, F_3) & g(F_3, F_0, F_2) & f_1(F_1, F_0, F_3) & g(F_1, F_0, F_2) \\ g(F_2, F_1, F_3) & h(F_0, F_3, F_2) & g(F_0, F_1, F_3) & h(F_0, F_1, F_2) \\ f_2(F_1, F_2, F_3) & g(F_3, F_0, F_2) & f_2(F_1, F_0, F_3) & g(F_1, F_0, F_2) \\ g(F_2, F_1, F_3) & h(F_0, F_3, F_2) & g(F_0, F_1, F_3) & h(F_0, F_1, F_2) \end{bmatrix}.$$

Consequently, showing that $\mathcal{J}_{\mathcal{A}}(F) \geq 0$ for $F \in \mathcal{E}$ is equivalent to showing that each of the functions $f_1(x, y, z)$, $f_2(x, y, z)$, $g(x, y, z)$, and $h(x, y, z)$ is nonnegative for $M_- = 0 \leq x, y, z \leq M_+ = \frac{1}{2}$. We now show that the local and boundary minima of f_1, f_2, g, and h are nonnegative. Examining the gradients of these functions, we find (1) $f_1(x, y, z) = 0$ at the point $x = y = z = \frac{1+\epsilon}{2}$, (2) $f_2(x, y, z) = 0$ at the point $x = y = z = \frac{1-\epsilon}{2}$, and (3) $g(x, y, z) = h(x, y, z) = 0$ at the point $x = y = z = \frac{1}{2}$. However, the Hessians $D^2 f_1$, $D^2 f_2$, $D^2 g$, and $D^2 h$ are nowhere positive or negative definite. Hence, there are no interior extrema.

Standard variational arguments on the boundary easily establish that

$$\begin{gathered} \min\{0, \tfrac{1+\epsilon}{2}, \tfrac{1+\epsilon}{2} - \tfrac{1}{4}\} \leq f_1 \leq \max\{\tfrac{1}{4}, \tfrac{1+\epsilon}{2}, \tfrac{1+\epsilon}{2} - \tfrac{1}{4}\}, \\ \min\{0, \tfrac{1-\epsilon}{2}, \tfrac{1-\epsilon}{2} - \tfrac{1}{4}\} \leq f_2 \leq \max\{\tfrac{1}{4}, \tfrac{1-\epsilon}{2}, \tfrac{1-\epsilon}{2} - \tfrac{1}{4}\}, \\ 0 \leq g, h \leq \tfrac{1}{2}. \end{gathered}$$

If $\epsilon \in [-\frac{1}{2}, \frac{1}{2}]$, it quickly follows that $0 \leq f_1, f_2 \leq \frac{1}{2}$. □

In LB3, Lemma 2.4 (invariance of the advection operator on domains of monotonicity) does not apply because the collision operator is nonzero at one of the extreme points of the domain of monotonicity $\mathcal{E}$. *Indeed,* $\mathcal{C}[F](\mathbf{M}_+) = \frac{\epsilon}{4}\mathbf{q}_1$, *which tends to zero as* ϵ *tends to zero, however.* Hence, in the limit as $\epsilon \to 0$ the invariance property is recovered. This suggests the possibility of invariance on $\mathcal{E}$ of the advection operator under conditions other than those of Lemma 2.4.

LEMMA 2.12. *Let* $\epsilon' \in (M_-, M_+)$ *be given. Let* $\mathcal{E}' = [M_-, M_+ - \epsilon']^{\mathrm{V}}$. *Then there exists* $\epsilon \in (-\epsilon', \epsilon')$ *such that if* $F(0, \mathrm{x}, \cdot) \in \mathcal{E}'$ *then* $F(m, \mathrm{x}, \cdot) \in \mathcal{E}$, *for all* m *such that* $0 < m\Delta t < T$, *where* Δt *and* ϵ *are defined in Theorem* 2.8.

Proof. We outline the proof as follows: (i) determine the extrema of $H(F)$ given that $F \in \mathcal{E}''$, where $\mathcal{E}'' = [M_-, M_+ - \zeta]$ and $\zeta \in (M_-, M_+)$; (ii) choose ϵ (depending on ϵ', M_-, M_+, T, K, and L, and the extrema determined in (i)) so that $H(F)$ applied recursively m times to $F(0, \mathbf{i}, \cdot) \in \mathcal{E}'$ yields a result in $\mathcal{E}$; and (iii) turn the recursion in step (ii) into an induction on m. □

The rather severe weaknesses of Lemma 2.12 suggest that LB3 may have difficulties in practice, as will be evident in the numerical studies in the subsequent section.

2.4.3. Summary. Given the above theorems, approximate solutions to (1) can be obtained from LB1, LB2, and LB3 according to the following prescription. Define $u_{\mathbf{i}}^m = \frac{1}{4}\sum_{\mathrm{v}\in\mathrm{V}} F(m,\mathbf{i},\mathrm{v})$ and $p_{\mathbf{i}}^m = cA'(u_{\mathbf{i}}^m)$, with $u_{\mathbf{i}}^m$ being defined as per the local conservation law. Then in the ℓ_1 norm, $p_{\mathbf{i}}^m$ is a second-order spatial, first-order temporal approximation to $\rho(m\Delta t, \mathbf{i}\Delta x)$.

We have shown that under certain conditions and transformations the lattice Boltzmann methods LB1, LB2, and LB3 converge second order in space and first order in time to the solution of (1). Note that the convergence of LB3 appears to be somewhat weaker than that of LB1 and LB2. In part this may be due to the increased number of nonzero coefficients (of the eigenvectors of the linearized collision operator) of the fourth-order term in the Hilbert expansion for LB3, as compared to the corresponding terms in the expansions for LB1 and LB2. In particular, $\sigma^{(3)} = 0$ is a solution to the fifth-order consistency condition for both LB1 and LB2, while it is not a solution to the same for LB3. Hence, $c_k^{(4)}$, $k = 0, 1, 2, 3$, are nonzero in LB3, which strengthens the influence of the fourth-order term of the Hilbert expansion on the truncation error.

3. Numerical studies. Several numerical studies, motivated by an earlier work [6], were conducted for the purposes of verifying the theory, comparing the lattice Boltzmann methods with each other, and comparing the lattice Boltzmann and finite difference methods. For each of the lattice Boltzmann and finite difference methods, the studies included acquiring (numerical) error and timing measurements in order to investigate their accuracy, order of convergence, and performance. All the methods were implemented in FORTRAN77 on a Fujitsu VPX240/10 and on one processing element of a Fujitsu VPP500, which we denote the VPP500/1. Both systems comprise high-performance vector-processing architectures, which are discussed later in more detail.

The following test problem was used: solve (1) with $(x, y) \in \Omega = [0, L] \times [0, L]$, $L = 1$, and $t \in [0, T]$, $T = 1$, for periodic boundary conditions with diffusion coefficients $\nu \in \{2^{-k} : k = 2, 3, 4, 5, 6\}$. Accuracy data in the ℓ_1 and ℓ_∞ norms were collected for the time steps corresponding to $t \in \{2^{-k} : k = 5, 4, 3, 2\}$. (Due to space limitations we present only ℓ_∞ norm results for $\nu \in \{2^{-k} : k = 2, 4, 6\}$.)

3.1. Lattice Boltzmann implementations. The lattice Boltzmann methods LB1, LB2, and LB3 compute the occupation numbers $F_{i,j}^m(\mathrm{v}_0)$, $F_{i,j}^m(\mathrm{v}_1)$, $F_{i,j}^m(\mathrm{v}_2)$, and $F_{i,j}^m(\mathrm{v}_3)$ over the prescribed lattice locations (i, j) and time steps m. In addition to this computation, there is also the calculation of the quantity $u_{i,j}^m = \frac{1}{4}\sum_{\mathrm{v}\in\mathrm{V}} F_{i,j}^m(\mathrm{v})$, which is the lattice Boltzmann methods' approximation to the solution of the second-order consistency conditions (11), (14), (17) for LB1, LB2, and LB3, respectively. The approximations $u_{i,j}^m$ must further be transformed via $p_{i,j}^m = c(1 - 2u_{i,j}^m)$ to obtain the approximation $p_{i,j}^m$ to the solution of (1). (These extra calculations were not included when collecting the performance and timing measurements of the lattice Boltzmann methods, as they are needed only when sampling a particular time step, which is generally sparse compared to the number of time steps.)

The difference formulae for the lattice Boltzmann methods are listed in Table 5, and except for the treatment of the periodic boundary conditions, their kernel implementations are found in Table 6. The methods' arithmetic operation counts are given in Table 7.

3.2. The finite difference method. The finite difference method employed for solving (1) is the following conservative, forward Euler-type, central difference scheme:

$$\begin{aligned} P_{i,j}^{m+1} = {} & P_{i,j}^m - \tfrac{\Delta t}{4\Delta x}\left[(P_{i+1,j}^m)^2 - (P_{i-1,j}^m)^2\right] \\ & + \tfrac{\nu\Delta t}{(\Delta x)^2}\left[P_{i+1,j}^m - 2P_{i,j}^m + P_{i-1,j}^m\right] + \tfrac{\nu\Delta t}{(\Delta y)^2}\left[P_{i,j+1}^m - 2P_{i,j}^m + P_{i,j-1}^m\right]. \end{aligned} \tag{21}$$

TABLE 5

Lattice Boltzmann difference equations. Note that for brevity here $F^{m}_{i,j}(\mathrm{v}_k) \equiv F(m,(i,j),\mathrm{v}_k)$.

LB1
$F^{m+1}_{i-1,j}(\mathrm{v}_0) \;=\; \frac{1+\epsilon}{4}[F^{m}_{i,j}(\mathrm{v}_0)+F^{m}_{i,j}(\mathrm{v}_1)+F^{m}_{i,j}(\mathrm{v}_2)+F^{m}_{i,j}(\mathrm{v}_3)]$
$-\frac{\epsilon}{6}[(F^{m}_{i,j}(\mathrm{v}_0)+F^{m}_{i,j}(\mathrm{v}_2))(F^{m}_{i,j}(\mathrm{v}_1)+F^{m}_{i,j}(\mathrm{v}_3))+F^{m}_{i,j}(\mathrm{v}_0)F^{m}_{i,j}(\mathrm{v}_2)+F^{m}_{i,j}(\mathrm{v}_1)F^{m}_{i,j}(\mathrm{v}_3)]$
$F^{m+1}_{i,j-1}(\mathrm{v}_1) \;=\; \frac{1}{4}[F^{m}_{i,j}(\mathrm{v}_0)+F^{m}_{i,j}(\mathrm{v}_1)+F^{m}_{i,j}(\mathrm{v}_2)+F^{m}_{i,j}(\mathrm{v}_3)]$
$F^{m+1}_{i+1,j}(\mathrm{v}_2) \;=\; \frac{1-\epsilon}{4}[F^{m}_{i,j}(\mathrm{v}_0)+F^{m}_{i,j}(\mathrm{v}_1)+F^{m}_{i,j}(\mathrm{v}_2)+F^{m}_{i,j}(\mathrm{v}_3)]$
$+\frac{\epsilon}{6}[(F^{m}_{i,j}(\mathrm{v}_0)+F^{m}_{i,j}(\mathrm{v}_2))(F^{m}_{i,j}(\mathrm{v}_1)+F^{m}_{i,j}(\mathrm{v}_3))+F^{m}_{i,j}(\mathrm{v}_0)F^{m}_{i,j}(\mathrm{v}_2)+F^{m}_{i,j}(\mathrm{v}_1)F^{m}_{i,j}(\mathrm{v}_3)]$
$F^{m+1}_{i,j+1}(\mathrm{v}_3) \;=\; \frac{1}{4}[F^{m}_{i,j}(\mathrm{v}_0)+F^{m}_{i,j}(\mathrm{v}_1)+F^{m}_{i,j}(\mathrm{v}_2)+F^{m}_{i,j}(\mathrm{v}_3)]$

LB2
$F^{m+1}_{i-1,j}(\mathrm{v}_0) \;=\; \frac{1+\epsilon}{4}[F^{m}_{i,j}(\mathrm{v}_0)+F^{m}_{i,j}(\mathrm{v}_1)+F^{m}_{i,j}(\mathrm{v}_2)+F^{m}_{i,j}(\mathrm{v}_3)]$
$-\frac{\epsilon}{2}[F^{m}_{i,j}(\mathrm{v}_0)F^{m}_{i,j}(\mathrm{v}_2)+F^{m}_{i,j}(\mathrm{v}_1)F^{m}_{i,j}(\mathrm{v}_3)]$
$F^{m+1}_{i,j-1}(\mathrm{v}_1) \;=\; \frac{1}{4}[F^{m}_{i,j}(\mathrm{v}_0)+F^{m}_{i,j}(\mathrm{v}_1)+F^{m}_{i,j}(\mathrm{v}_2)+F^{m}_{i,j}(\mathrm{v}_3)]$
$F^{m+1}_{i+1,j}(\mathrm{v}_2) \;=\; \frac{1-\epsilon}{4}[F^{m}_{i,j}(\mathrm{v}_0)+F^{m}_{i,j}(\mathrm{v}_1)+F^{m}_{i,j}(\mathrm{v}_2)+F^{m}_{i,j}(\mathrm{v}_3)]$
$+\frac{\epsilon}{2}[F^{m}_{i,j}(\mathrm{v}_0)F^{m}_{i,j}(\mathrm{v}_2)+F^{m}_{i,j}(\mathrm{v}_1)F^{m}_{i,j}(\mathrm{v}_3)]$
$F^{m+1}_{i,j+1}(\mathrm{v}_3) \;=\; \frac{1}{4}[F^{m}_{i,j}(\mathrm{v}_0)+F^{m}_{i,j}(\mathrm{v}_1)+F^{m}_{i,j}(\mathrm{v}_2)+F^{m}_{i,j}(\mathrm{v}_3)]$

LB3
$F^{m+1}_{i-1,j}(\mathrm{v}_0) \;=\; \frac{1+\epsilon}{2}[F^{m}_{i,j}(\mathrm{v}_0)+F^{m}_{i,j}(\mathrm{v}_2)]-\epsilon F^{m}_{i,j}(\mathrm{v}_0)F^{m}_{i,j}(\mathrm{v}_2)$
$+\Big[F^{m}_{i,j}(\mathrm{v}_0)F^{m}_{i,j}(\mathrm{v}_2)[F^{m}_{i,j}(\mathrm{v}_1)+F^{m}_{i,j}(\mathrm{v}_3)]+F^{m}_{i,j}(\mathrm{v}_1)F^{m}_{i,j}(\mathrm{v}_3)[1-F^{m}_{i,j}(\mathrm{v}_0)-F^{m}_{i,j}(\mathrm{v}_2)]\Big]$
$F^{m+1}_{i,j-1}(\mathrm{v}_1) \;=\; \frac{1}{2}[F^{m}_{i,j}(\mathrm{v}_1)+F^{m}_{i,j}(\mathrm{v}_3)]$
$-\Big[F^{m}_{i,j}(\mathrm{v}_0)F^{m}_{i,j}(\mathrm{v}_2)[F^{m}_{i,j}(\mathrm{v}_1)+F^{m}_{i,j}(\mathrm{v}_3)]+F^{m}_{i,j}(\mathrm{v}_1)F^{m}_{i,j}(\mathrm{v}_3)[1-F^{m}_{i,j}(\mathrm{v}_0)-F^{m}_{i,j}(\mathrm{v}_2)]\Big]$
$F^{m+1}_{i+1,j}(\mathrm{v}_2) \;=\; \frac{1-\epsilon}{2}[F^{m}_{i,j}(\mathrm{v}_0)+F^{m}_{i,j}(\mathrm{v}_2)]+\epsilon F^{m}_{i,j}(\mathrm{v}_0)F^{m}_{i,j}(\mathrm{v}_2)$
$+\Big[F^{m}_{i,j}(\mathrm{v}_0)F^{m}_{i,j}(\mathrm{v}_2)[F^{m}_{i,j}(\mathrm{v}_1)+F^{m}_{i,j}(\mathrm{v}_3)]+F^{m}_{i,j}(\mathrm{v}_1)F^{m}_{i,j}(\mathrm{v}_3)[1-F^{m}_{i,j}(\mathrm{v}_0)-F^{m}_{i,j}(\mathrm{v}_2)]\Big]$
$F^{m+1}_{i,j+1}(\mathrm{v}_3) \;=\; \frac{1}{2}[F^{m}_{i,j}(\mathrm{v}_1)+F^{m}_{i,j}(\mathrm{v}_3)]$
$-\Big[F^{m}_{i,j}(\mathrm{v}_0)F^{m}_{i,j}(\mathrm{v}_2)[F^{m}_{i,j}(\mathrm{v}_1)+F^{m}_{i,j}(\mathrm{v}_3)]+F^{m}_{i,j}(\mathrm{v}_1)F^{m}_{i,j}(\mathrm{v}_3)[1-F^{m}_{i,j}(\mathrm{v}_0)-F^{m}_{i,j}(\mathrm{v}_2)]\Big]$

One can show, e.g., by techniques for monotone methods (see, e.g., [12, Chap. IV]), that (21) is an $\mathcal{O}\big[(\Delta x)^2\big]+\mathcal{O}\big[(\Delta y)^2\big]+\mathcal{O}[\Delta t]$ convergent, conservative, monotone finite difference method for (1). The monotonicity criteria that ensure stability are that $\Delta t \leq \frac{(\Delta x)^2}{4\nu}$ and $\|P\|_{\ell_\infty} \leq \frac{2\nu}{\Delta x}$ (the latter is analogous to a Courant condition, e.g., [11]).

The finite difference method (21) was implemented at its limit of stability, $\Delta t = \frac{(\Delta x)^2}{4\nu}$, which is inherent in the lattice Boltzmann methods LB1, LB2, and LB3. Thus, the finite difference and lattice Boltzmann methods are approximating solutions to (1) for identical discretizations of the spatial–temporal domain ($[0, L] \times [0, L]) \times [0, T)$ and can be straightforwardly compared. Barring the treatment of the periodic boundary conditions, the implementation of the finite difference kernel is given in Table 6. Its arithmetic operation count is also given in Table 7.

TABLE 6
Kernel implementations of the lattice Boltzmann and finite difference methods.

LB1	LB3
for $i = 2$ to $N-1$ for $j = 2$ to $N-1$ $S_{02} = F_{i,j}(\mathrm{v}_0) + F_{i,j}(\mathrm{v}_2)$ $P_{02} = F_{i,j}(\mathrm{v}_0) \times F_{i,j}(\mathrm{v}_2)$ $S_{13} = F_{i,j}(\mathrm{v}_1) + F_{i,j}(\mathrm{v}_3)$ $P_{13} = F_{i,j}(\mathrm{v}_1) \times F_{i,j}(\mathrm{v}_3)$ $S_{0123} = S_{02} + S_{13}$ $\rho = \frac{1}{4} \times S_{0123}$ $T_1 = P_{02} + P_{13}$ $T_2 = S_{02} \times S_{13}$ $T_3 = \frac{1-2a}{6} \times (T_1 + T_2)$ $F'_{i+1,j}(\mathrm{v}_0) = \frac{a}{2} \times S_{0123} + T_3$ $F'_{i,j+1}(\mathrm{v}_1) = \rho$ $F'_{i-1,j}(\mathrm{v}_2) = \frac{1-a}{2} \times S_{0123} - T_3$ $F'_{i,j-1}(\mathrm{v}_3) = \rho$ end for end for	for $i = 2$ to $N-1$ for $j = 2$ to $N-1$ $S_{02} = F_{i,j}(\mathrm{v}_0) + F_{i,j}(\mathrm{v}_2)$ $P_{02} = F_{i,j}(\mathrm{v}_0) \times F_{i,j}(\mathrm{v}_2)$ $S_{13} = F_{i,j}(\mathrm{v}_1) + F_{i,j}(\mathrm{v}_3)$ $P_{13} = F_{i,j}(\mathrm{v}_1) \times F_{i,j}(\mathrm{v}_3)$ $T_1 = P_{02} \times (S_{13} - 1)$ $T_2 = P_{13} \times (S_{02} - 1)$ $T_3 = T_1 - T_2$ $T_4 = \frac{1}{2} \times S_{13} - T_3$ $T_5 = (1 - 2a) \times P_{02}$ $T_6 = a \times S_{02}$ $T_7 = T_5 + T_6$ $F'_{i+1,j}(\mathrm{v}_0) = T_3 + T_7$ $F'_{i,j+1}(\mathrm{v}_1) = T_4$ $F'_{i-1,j}(\mathrm{v}_2) = S_{02} + T_3 - T_7$ $F'_{i,j-1}(\mathrm{v}_3) = T_4$ end for end for
LB2	Finite Difference

for $i = 2$ to $N-1$
for $j = 2$ to $N-1$
$S_{02} = F_{i,j}(\mathrm{v}_0) + F_{i,j}(\mathrm{v}_2)$
$P_{02} = F_{i,j}(\mathrm{v}_0) \times F_{i,j}(\mathrm{v}_2)$
$S_{13} = F_{i,j}(\mathrm{v}_1) + F_{i,j}(\mathrm{v}_3)$
$P_{13} = F_{i,j}(\mathrm{v}_1) \times F_{i,j}(\mathrm{v}_3)$
$S_{0123} = S_{02} + S_{13}$
$\rho = \frac{1}{4} \times S_{0123}$
$T_1 = P_{02} + P_{13}$
$T_2 = (\frac{1}{4} - a) \times T_1$
$F'_{i+1,j}(\mathrm{v}_0) = \frac{a}{2} \times S_{0123} + T_2$
$F'_{i,j+1}(\mathrm{v}_1) = \rho$
$F'_{i-1,j}(\mathrm{v}_2) = \frac{1-a}{2} \times S_{0123} - T_2$
$F'_{i,j-1}(\mathrm{v}_3) = \rho$
end for
end for
	for $i = 2$ to $N-1$
	for $j = 2$ to $N-1$
	$S_x = P^m_{i+1,j} + P^m_{i-1,j}$
	$S_y = P^m_{i,j+1} + P^m_{i,j-1}$
	$T = \frac{\Delta t}{4\Delta x} \times (P^m_{i+1,j} - P^m_{i-1,j})$
	$P^{m+1}_{i,j} = \frac{1}{4} \times S_y + S_x \times (T + \frac{1}{4})$
	end for
	end for

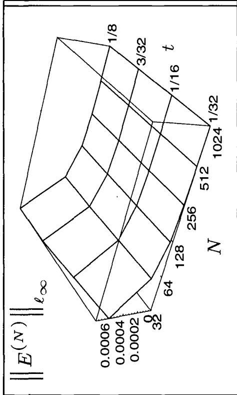
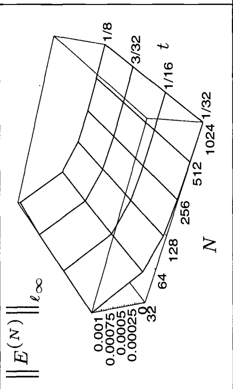
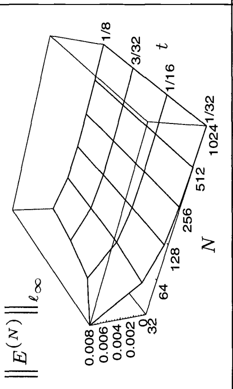
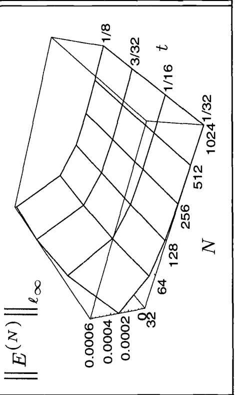
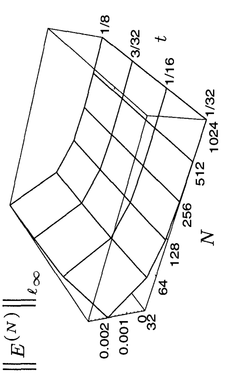
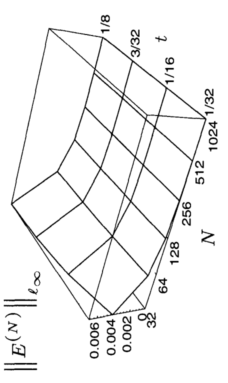
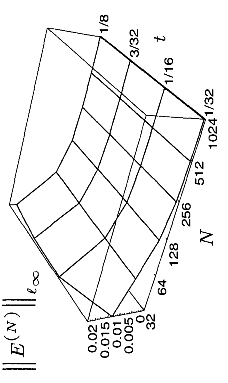
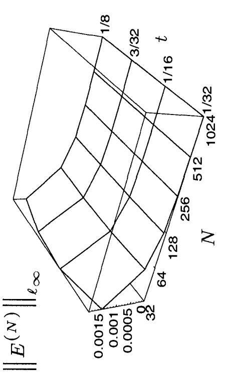
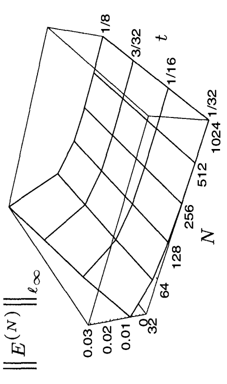
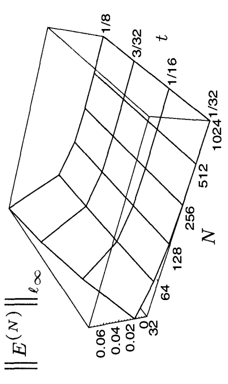
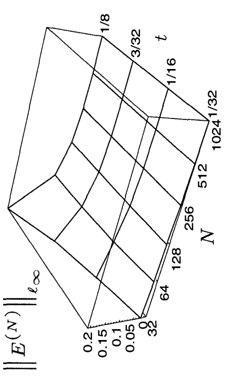
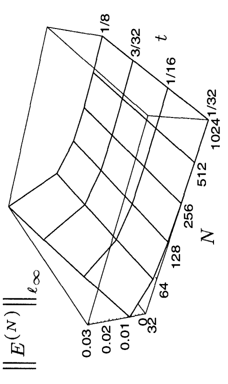
TABLE 7
Floating point operation (flop) counts.

METHOD	OPERATION COUNT		
	Additions/Subtractions (flop)	Multiplications (flop)	Total (flop)
LB1	7	7	14
LB2	6	6	12
LB3	10	7	17
FD	5	3	8

3.3. Accuracy. The absolute errors in the ℓ_∞ norm for LB1, LB2, LB3, and the finite difference method are given in Table 8. We see that the absolute errors worsen progressively from LB1 to LB2 to LB3, with the errors in LB3 being significantly worse than in LB1. We also see that all the methods depict increasing accuracy with increasing diffusion coefficients. The finite difference method reflects the greatest accuracy of all the methods, however, with LB1 being a close second. The ratios of the errors observed in LB1, LB2, and LB3 to those

TABLE 8

Accuracy. Absolute errors in the ℓ_∞ norm of the lattice–Boltzmann-computed and finite–difference-computed solutions are given for various diffusion coefficients ν.

ν	LB1	LB2	LB3	Finite Difference
1/4				
1/16				
1/64				

in the finite difference method appear in Table 9. Overall the errors in LB1 are quite good compared to those in the finite difference method, with the maximum error observed being about twice as great. The results for LB3 are not as good, with the errors being as high as about 100 times those in the finite difference method. (Similar comments apply to the errors in the ℓ_1 norm.)

TABLE 9

Error comparisons. Ratios between lattice–Boltzmann-computed and finite–difference-computed solutions are given for various diffusion coefficients ν.

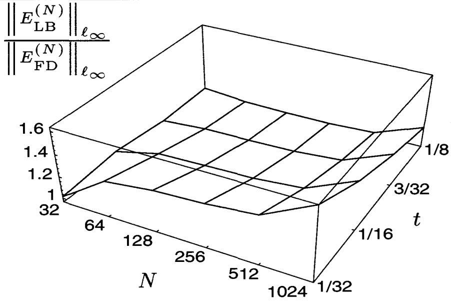
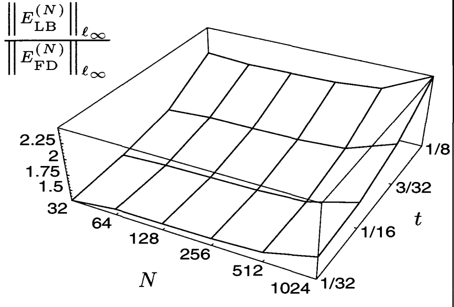
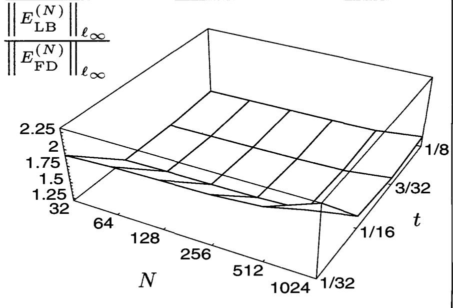
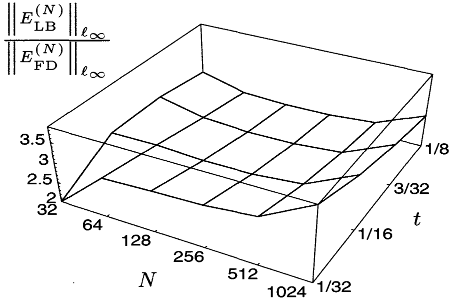
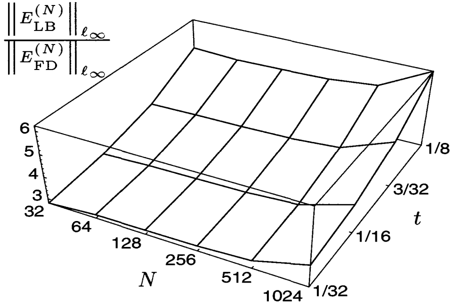
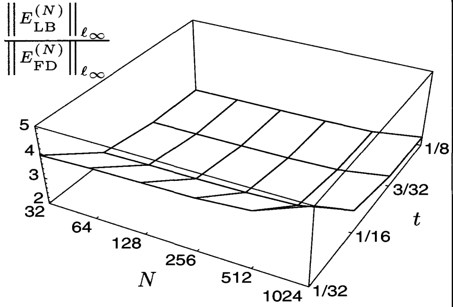
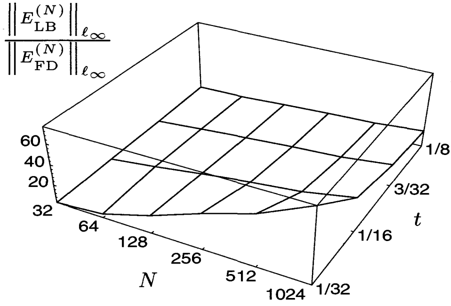
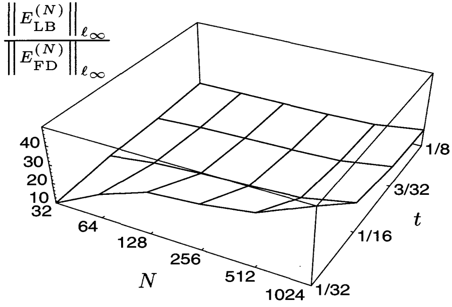
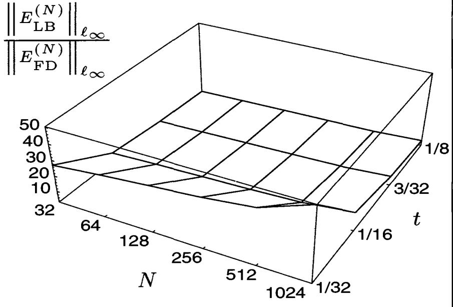
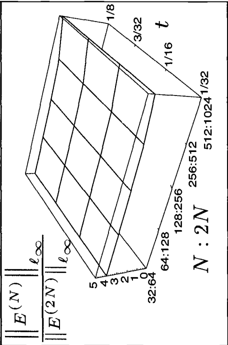
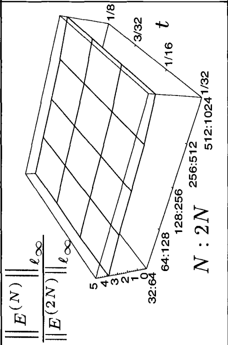
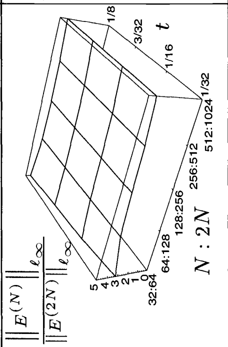
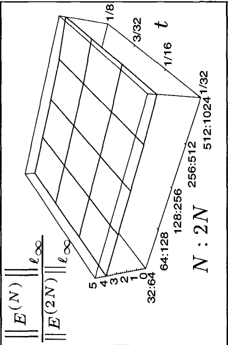
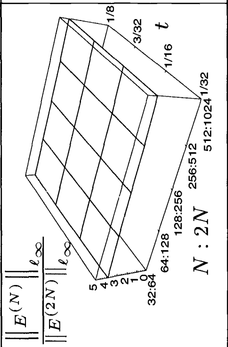
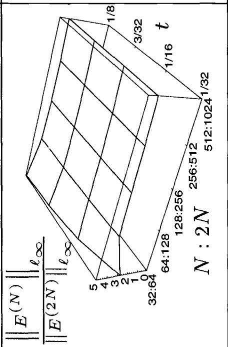
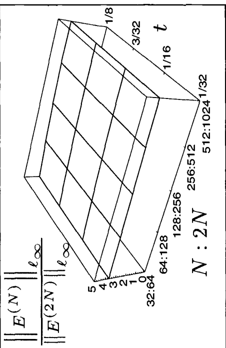
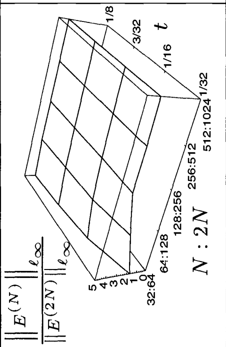
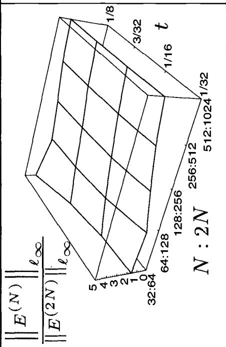
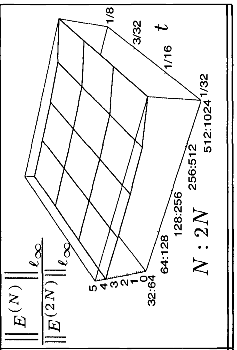
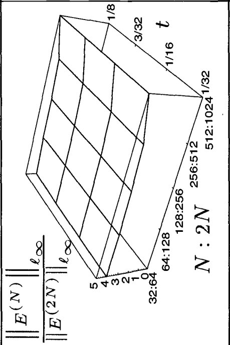
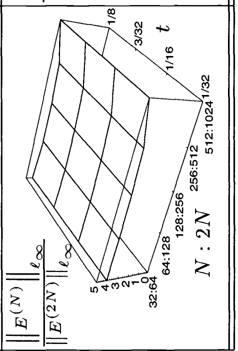
ν	1/64	1/16	1/4
LB1			
LB2			
LB3			

TABLE 10

Order of convergence. This charts the ratios between the errors for solutions computed on $N \times N$ and $(2N) \times (2N)$ grids.

ν	1/4	1/16	1/64
LB1			
LB2			
LB3			
Finite Difference			

3.4. Order of convergence. The ratios between the absolute errors at a given problem size and twice that problem size, i.e., for $N \times N$ and $(2N) \times (2N)$ grids, respectively, can be reflective of the order of convergence of a supposed second-order convergent method. Although not necessary for second-order convergence, a ratio near four is indictive of second-order convergence. The error ratios in the ℓ_∞ norm for LB1, LB2, LB3, and the finite difference method are given in Table 10.

In the finite difference method, the error ratio is definitely near four, which supports the theoretical second-order convergence of the method. The error ratio in LB3 appears slightly less than four, which mimics its difficulties with accuracy, as pointed out in §2.4. Across the lattice Boltzmann methods LB3, LB2, and LB1, the error ratio progressively approaches four. In addition, the lattice Boltzmann methods appear generally to produce error ratios nearer to four for larger diffusion coefficients. This is expected, however, since the convergence proofs assume that $\epsilon = \frac{\Delta x}{2\nu} = \mathcal{O}[\Delta x]$. So, for a particular lattice size, i.e., a fixed Δx, the smaller the diffusion coefficient ν, the less ϵ appears to be $\mathcal{O}[\Delta x]$. Indeed, for $\nu = \mathcal{O}[\Delta x]$ the convection bias ϵ is $\mathcal{O}[1]$ and the convergence proofs for LB1, LB2, and LB3 fail altogether. As the theory predicts, this behavior is evident in the error ratios shown. Of the three lattice Boltzmann methods, LB1 appears to exhibit the most convincing error ratios for the widest variety of diffusion coefficients. (Again, the above comments apply to the ℓ_1 norm also.)

3.5. Performance and timing measurements. Because the errors in LB1 are close to those in the finite difference method (they are at most a factor of two apart), LB1 might be considered an alternative to the finite difference method in some circumstances, which among other considerations may depend on the performance of the methods on a particular architecture. To this end, we obtained performance and timing measurements for the three lattice Boltzmann methods and the finite difference method for two high-performance computer systems: a Fujitsu VPX240/10, which has a peak performance of 2.5 Gflop/s, and a Fujitsu VPP500/1, one processing element (PE) of which has a peak performance of 1.6 Gflop/s. While both systems comprise vector-pipelined architectures, they differ in several ways.

1. The VPX240/10 has two multiply–add pipelines, each of which delivers two results per clock cycle; a VPP500 PE has one multiply pipeline and one add pipeline, each of which delivers eight results per clock cycle. In both systems, the vector pipelines can operate concurrently and can also be chained.
2. The VPX240/10 has two universal load/store pipelines, each of which can operate concurrently; a VPP500 PE has one load pipeline and one store pipeline, each of which can operate concurrently. In both systems the load/store and arithmetic pipelines can operate concurrently.
3. The VPP500 PEs and the VPX240/10 possess 32-way and 256-way interleaved memory, respectively.
4. A VPP500 PE has a larger vector register size (128 kilobytes) than does the VPX240/10 (64 kilobytes). Both vector register sets are reconfigurable: from 8×2048-element to 256×64-element 64-bit vectors on the VPP500/1 and from 8×1024-element to 256×32-element 64-bit vectors on the VPX240/10.

Further characteristics of these systems can be found in [13–15]. The above identifying characteristics affect the way the various algorithms perform on the two systems. For instance, because the arithmetic vector pipelines on the VPX240/10 are multiply–add pipelines, peak performance is achieved with two vector "multiply–add" instructions operating simultaneously on the separate pipelines; concurrent "add" and "multiply" instructions achieve the peak performance on the VPP500. Which architecture performs best for a given application depends, among other considerations, on how well vectorizable loops can be mapped to the peak performance situations, as we will see.

3.5.1. Theory. By analyzing the corresponding dependency graphs of our methods, the performance of the applications on the two architectures can be predicted. Given a particular architecture and assuming a certain level of unrolling, a dependency graph can be mapped to a timing diagram, which then can be used to predict how the calculation will perform. In our case, the architectures are vector-pipelined machines. Thus, the timing diagrams will involve scheduling of the pipelines into chimes, as the arithmetic and load/store pipelines can be chained.

As examples, we will examine closely the LB1 and finite difference calculations and only summarize the results for LB2 and LB3. Figures 2 and 3 depict the dependency graphs for LB1 and the finite difference method, respectively (cf. Table 6). Mapping the dependency graphs onto the two architectures in an optimal fashion (without loop unrolling) yields the pipeline

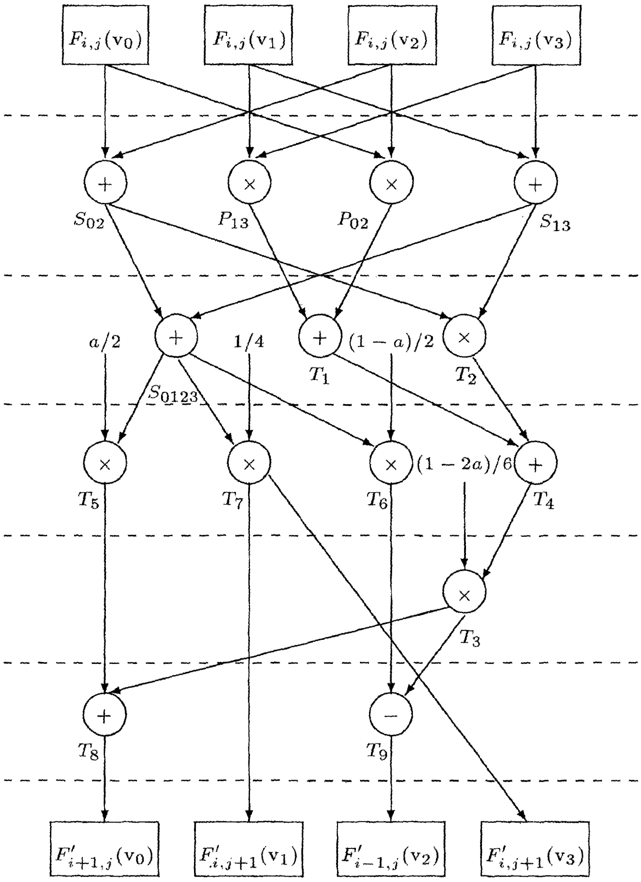


FIG. 2. *Dependency graph of LB*1 *computation.*

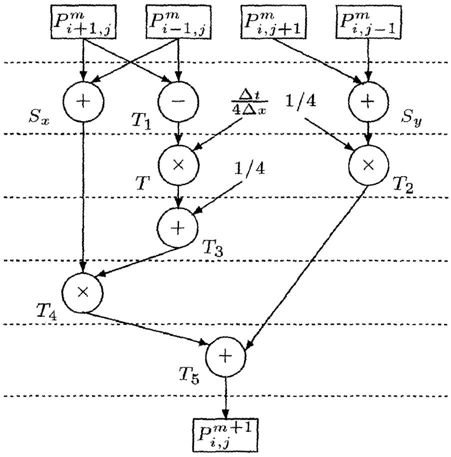


FIG. 3. *Dependency graph of finite difference computation*

scheduling timing diagrams given in Figures 4 and 5. Note that a certain amount of unrolling can offset some of the load and store operations and occupy otherwise idle arithmetic pipelines. Table 11 summarizes the chimes to compute the lattice Boltzmann and finite difference kernels for the two vector architectures in the study. With regard to timing predictions, Table 11 suggests certain relative timings for the methods. Table 12 gives the asymptotic timings of each lattice Boltzmann method relative to the finite difference method.[4] The predictions and actual results are compared below.

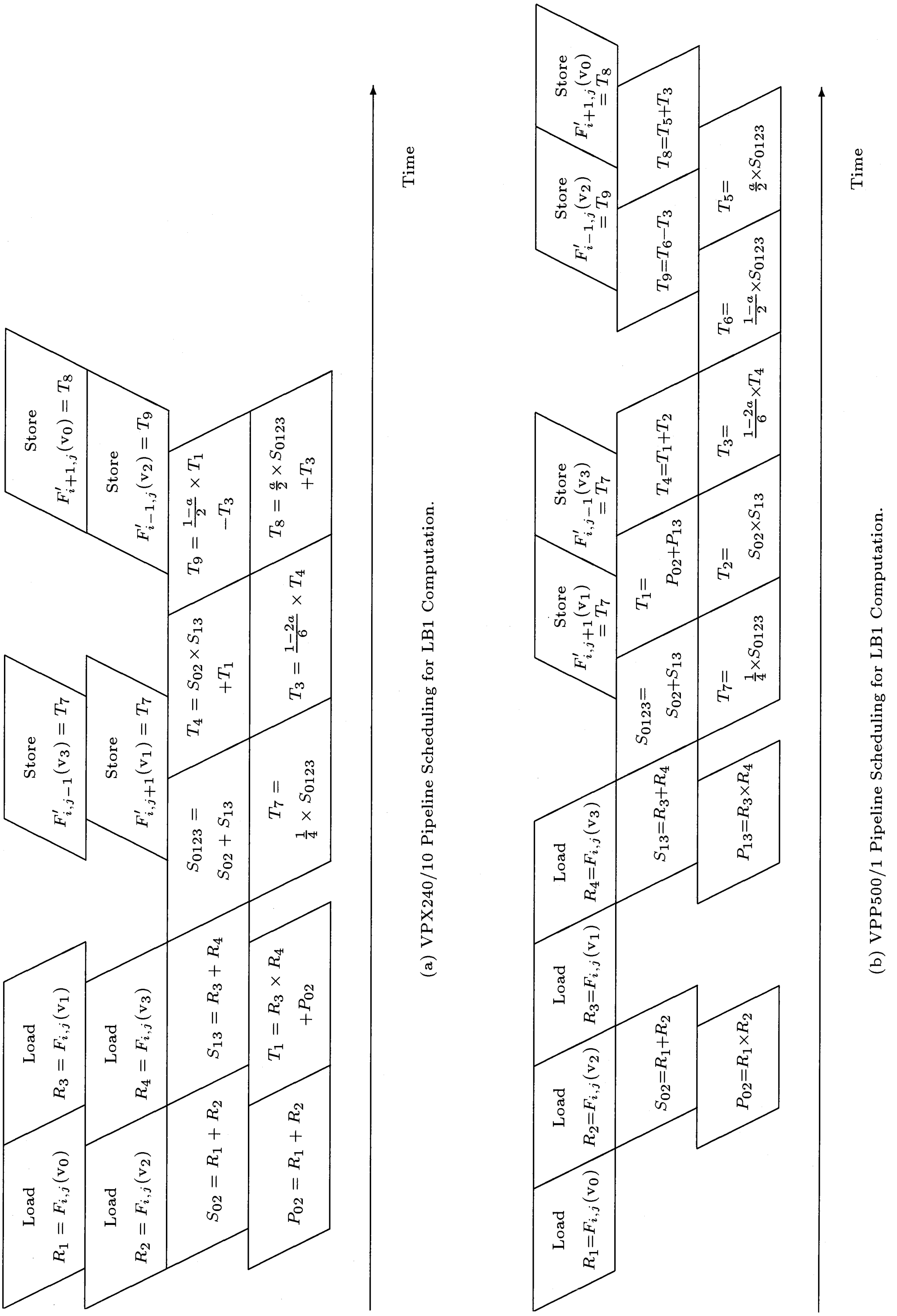
3.5.2. Measurements. Performance and timing measurements were obtained on the kernels for all four methods. These measurements included both interior and boundary node calculations. Measurements were taken for 512 time steps and a range of problem sizes $N \times N$ up to 1617×1617. Tables 13 and 14 list figures exhibiting the performance and timing measurements for the VPX240/10 and the VPP500/1, respectively. Timings relative to that of the finite difference method are depicted in Table 15. Asymptotically, we see that, respectively, LB1, LB2, and LB3 consume about 1.5, 1.4, and 2.0 of the time required of the finite difference computation, which is in agreement with the predictions of Table 12.

In terms of absolute performance, all four of the methods attain their highest performance levels on the VPX240/10 (Table 16). The highest maximum utilization (91.8%) is with LB1 on the VPP500/1; the lowest maximum utilization (53.8%) is with the finite difference method on the VPX240/10. Accuracy aside, a bottom-line comparison considers how much time it takes to compute a solution for a given problem size. Because of the differences observed in the utilization of the two systems, we find that the computational time for both systems is nearly the same for each of the respective methods, except for problem sizes below about 100×100. For each of the lattice Boltzmann and finite difference methods, the ratio between the VPP500/1 and VPX240/10 computational times is exhibited in the last column of Table 14. Indeed, for problems larger than about 100×100, the computational time on the VPP500/1 is less than 120% of that on the VPX240/10. This is interesting to note because the peak performance of the VPP500/1 is only 64% that of the VPX240/10, and it accents differences between the two vector-pipelined architectures.

3.6. Memory requirements. The memory requirements (in bytes) using 64-bit data for the lattice Boltzmann and finite difference methods are given by $M_{\text{lb}}(N_x, N_y) = 64N_xN_y$ and $M_{\text{fd}}(N_x, N_y) = 16N_xN_y$, respectively. However, due to the exclusion principle of the lattice Boltzmann methods, it may not be necessary to maintain full 64-bit precision arithmetic. Using single-bit precision arithmetic (with a special rounding function) would be analogous to a lattice gas method. But with regard to solving partial differential equations, lattice gas methods suffer from statistical fluctuations due to the limited precision of the occupation numbers. Certainly, there is some trade-off here between the full precision of lattice Boltzmann methods and the extremely limited precision of lattice gas methods. (See, for example, [2] for a thorough treatment of correlations in lattice gas methods.)

3.7. A remark on conservation laws and accuracy. The one conservation law in the lattice Boltzmann methods LB1, LB2, and LB3 is conservation of mass. Although they each possess (only) this local conservation law, the individual collision rules therein do not wholly adhere to it. Consider, for example, the single particle collision rules for LB3 (see Table 4). The mass along each of the two orthogonal directions is conserved separately. (A

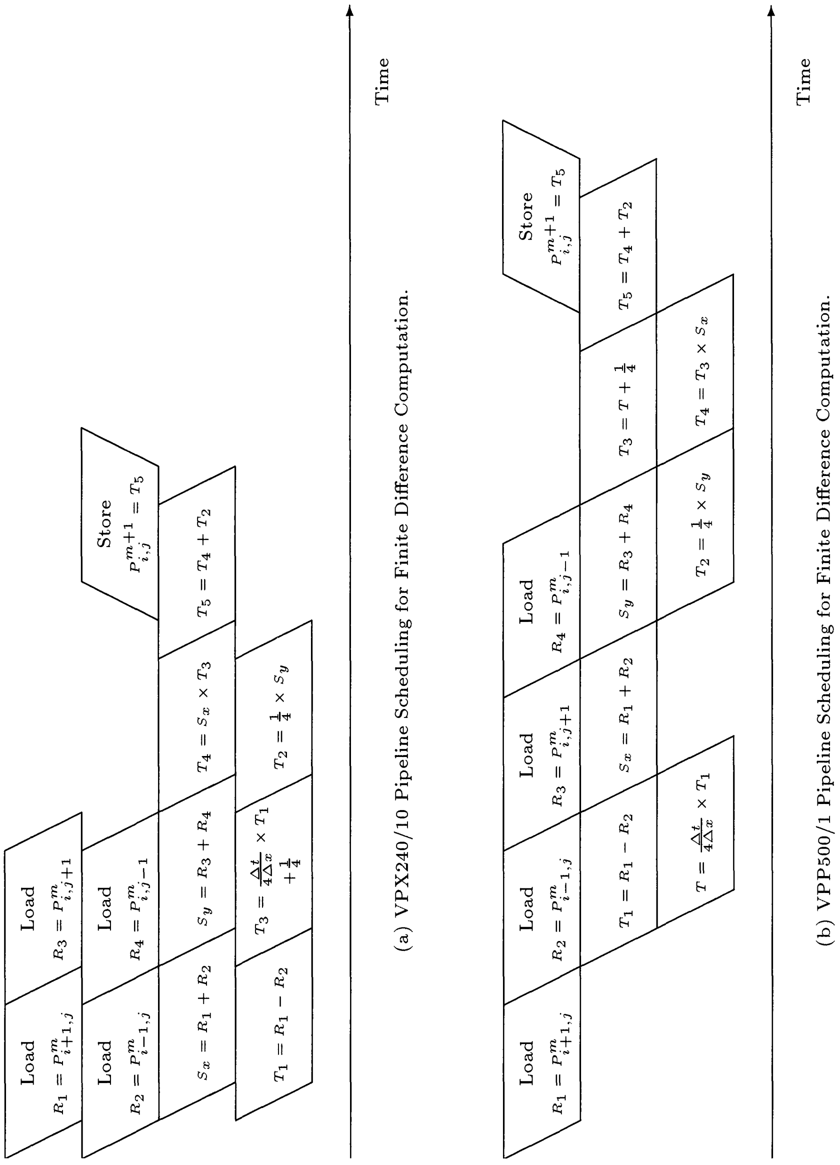
[4] Note that in some cases, increasing the depth of unrolling can yield a *slight* gain in performance, but this is not significant with regard to our discussions.


Load $R_1 = F_{i,j}(v_0)$
Load $R_3 = F_{i,j}(v_1)$
Store $F'_{i,j-1}(v_3) = T_7$
Store $F'_{i+1,j}(v_0) = T_8$
Load $R_2 = F_{i,j}(v_2)$
Load $R_4 = F_{i,j}(v_3)$
Store $F'_{i,j+1}(v_1) = T_7$
Store $F'_{i-1,j}(v_2) = T_9$
$S_{02} = R_1 + R_2$
$S_{13} = R_3 + R_4$
$S_{0123} = S_{02} + S_{13}$
$T_4 = S_{02} \times S_{13} + T_1$
$T_9 = \frac{1-a}{2} \times T_1 - T_3$
$P_{02} = R_1 + R_2$
$T_1 = R_3 \times R_4 + P_{02}$
$T_7 = \frac{1}{4} \times S_{0123}$
$T_3 = \frac{1-2a}{6} \times T_4$
$T_8 = \frac{a}{2} \times S_{0123} + T_3$
Time
Load $R_1{=}F_{i,j}(v_0)$
Load $R_2{=}F_{i,j}(v_2)$
Load $R_3{=}F_{i,j}(v_1)$
Load $R_4{=}F_{i,j}(v_3)$
Store $F'_{i,j+1}(v_1) = T_7$
Store $F'_{i,j-1}(v_3) = T_7$
Store $F'_{i-1,j}(v_2) = T_9$
Store $F'_{i+1,j}(v_0) = T_8$
$S_{02}{=}R_1{+}R_2$
$S_{13}{=}R_3{+}R_4$
$S_{0123}{=} S_{02}{+}S_{13}$
$T_1{=} P_{02}{+}P_{13}$
$T_4{=}T_1{+}T_2$
$T_9{=}T_6{-}T_3$
$T_8{=}T_5{+}T_3$
$P_{02}{=}R_1{\times}R_2$
$P_{13}{=}R_3{\times}R_4$
$T_7{=} \frac{1}{4}{\times}S_{0123}$
$T_2{=} S_{02}{\times}S_{13}$
$T_3{=} \frac{1-2a}{6}{\times}T_4$
$T_6{=} \frac{1-a}{2}{\times}S_{0123}$
$T_5{=} \frac{a}{2}{\times}S_{0123}$
Time

(a) VPX240/10 Pipeline Scheduling for LB1 Computation.

(b) VPP500/1 Pipeline Scheduling for LB1 Computation.

FIG. 4. *Pipeline scheduling timing diagrams for the LB*1 *computation. Note that the calculations entail five* (5) *and nine* (9) *chimes on the* VPX240/10 *and* VPP500/1, *respectively.*



(a) VPX240/10 Pipeline Scheduling for Finite Difference Computation.

(b) VPP500/1 Pipeline Scheduling for Finite Difference Computation.

FIG. 5. *Pipeline scheduling timing diagrams for the finite difference computation. Note that the calculations entail four* (4) *and six* (6) *chimes on the* VPX240/10 *and* VPP500/1, *respectively.*

TABLE 11
Chimes to compute the lattice Boltzmann and finite difference kernels.

METHOD	VPX240/10			VPP500/1		
	Without Unrolling	As Implemented		Without Unrolling	As Implemented	
	Chimes (count)	Unroll (depth)	Chimes (average)	Chimes (count)	Unroll (depth)	Chimes (average)
LB1	5	1	5.00	9	4	7.50
LB2	5	2	5.00	8	2	7.00
LB3	7	2	7.00	12	9	10.22
FD	4	2	3.50	6	7	5.14

Table 12
Predicted asymptotic timings relative to the finite difference method.

METHOD	VPX240/10			VPP500/1		
	Without Unrolling	As Implemented		Without Unrolling	As Implemented	
	Time (FD time)	Unroll (depth)	Time (FD time)	Time (FD time)	Unroll (depth)	Time (FD time)
LB1	1.25	1	1.43	1.50	4	1.46
LB2	1.25	2	1.43	1.33	2	1.36
LB3	1.75	2	2.00	2.00	2	1.99
FD	1.00	2	1.00	1.00	7	1.00

Table 13
Performance measurements for the lattice Boltzmann and finite difference methods. The performance measurements were calculated by dividing computational time (CPU time) by a kernel's floating point operation count (boundary conditions included), the number of time steps, and the number of grid nodes. The entries chart performance versus size N for N × N grids.

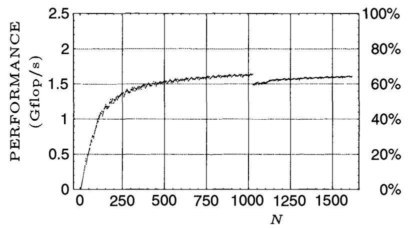
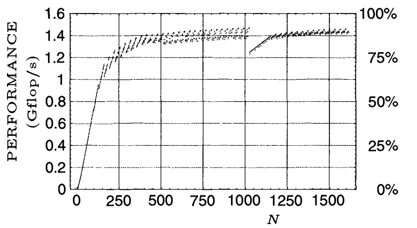
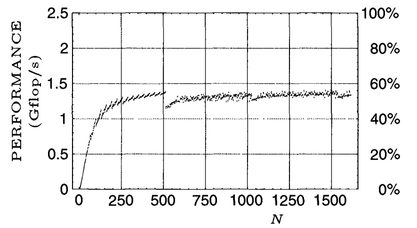
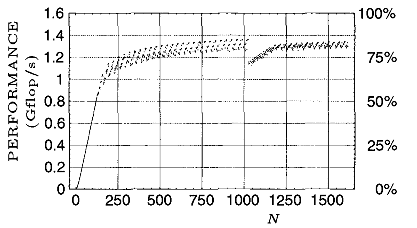
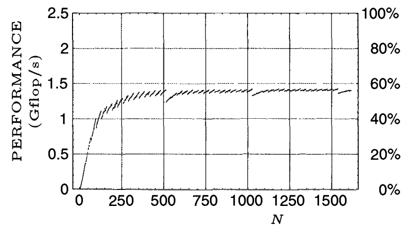
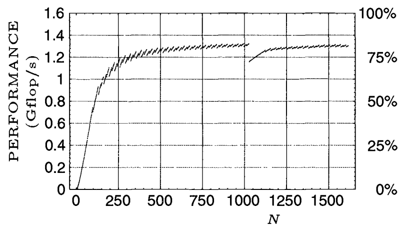
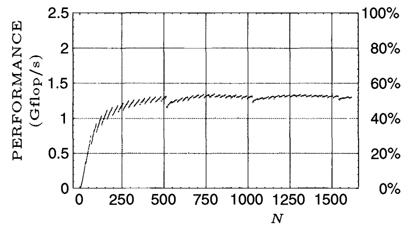
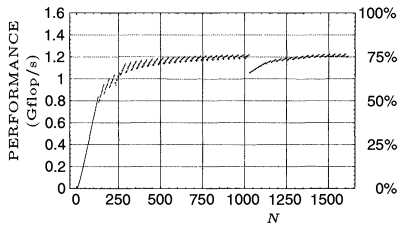
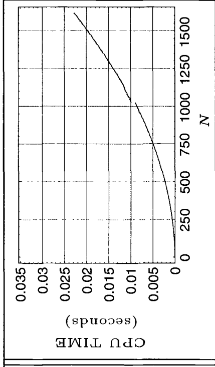
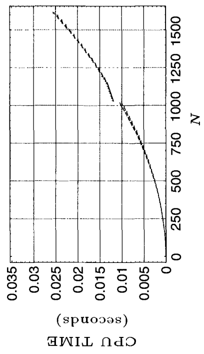
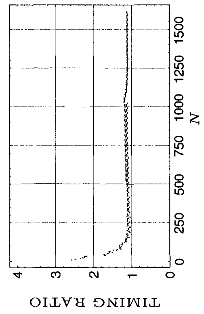
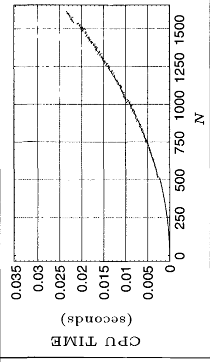
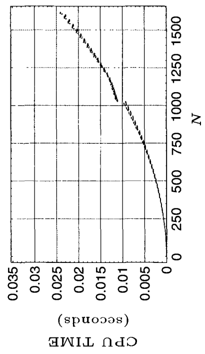
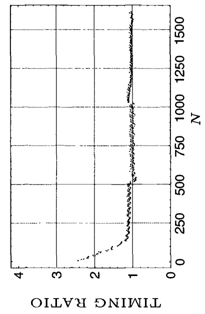
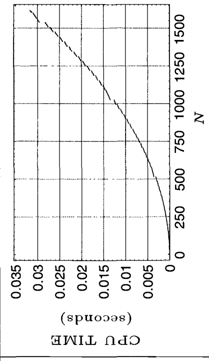
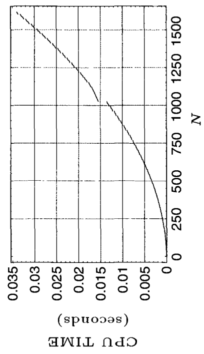
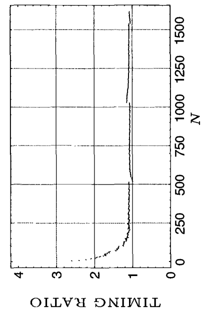
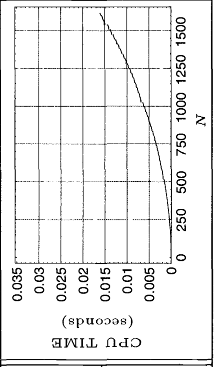
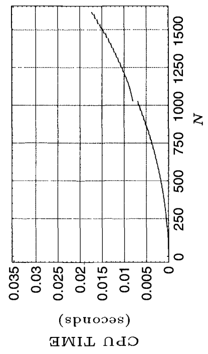
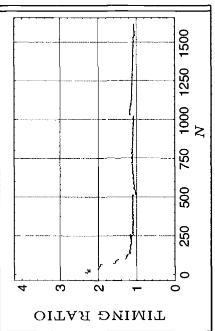
METHOD	PERFORMANCE MEASUREMENTS	
	VPX240/10 (2.5 Gflop/s peak)	VPP500/1 (1.6 Gflop/s peak)
LB1		
LB2		
LB3		
Finite Difference		

TABLE 14

Timing measurements and system comparisons for the lattice Boltzmann and finite difference methods. The computational time (CPU time) is given on a per-time-step basis as a function of the problem size N, where the grid is size N × N. All calculations were performed in 64*-bit precision. The ratios between the timings for the* VPP500/1 *and* VPX240/10 *systems are also given.*

METHOD	TIMING MEASUREMENTS VPX240/10 (2.5 Gflop/s peak)	TIMING MEASUREMENTS VPP500/1 (1.6 Gflop/s peak)	SYSTEM RATIOS $\left(\frac{\text{VPP500/1 time}}{\text{VPX240/10 time}}\right)$
LB1			
LB2			
LB3			
Finite Difference			

similar observation can be made about the triple particle collisions in LB3.) However, in LB2 the single- and triple-particle collision rules do not preserve mass along the two orthogonal directions separately. (This is also the case in LB1.) The difference between LB1 and LB2 is that in LB1 two-particle collisions involve all possible configurations, i.e., both "orthogonal" and "head-on" collisions, whereas in LB2 the orthogonal and head-on configurations are treated separately. Keeping all this in mind, it is interesting to note that the accuracy of the lattice Boltzmann methods improves progressively from LB3 to LB2 to LB1. This suggests the hypothesis that lattice Boltzmann methods achieve greater accuracy when the collision rules are individually faithful to the overriding local conservation laws.

Table 15

Lattice Boltzmann to finite difference timing comparisons. Ratios of lattice Boltzmann to finite difference computational time, based on Table 14, are given on a per-time-step basis as a function of problem size N, for N × N grids.

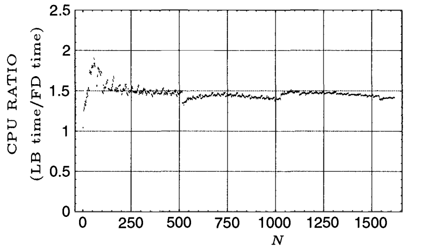
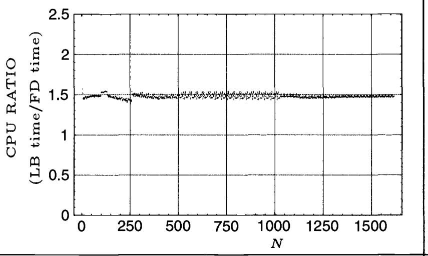
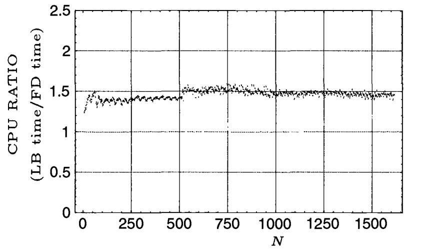
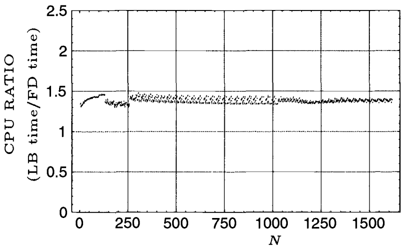
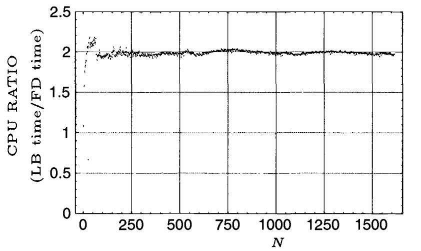
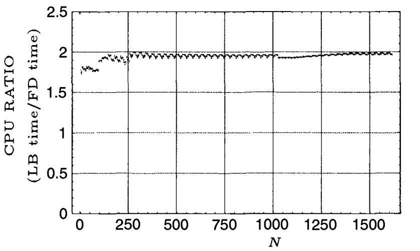
METHOD	RATIO TO FINITE DIFFERENCE COMPUTATION TIME	
	VPX240/10 (2.5 Gflop/s peak)	VPP500/1 (1.6 Gflop/s peak)
LB1		
LB2		
LB3		

Table 16

Maximum sustained performances.

METHOD	VPX240/10			VPP500/1		
	Size (N×N)	Maximum Performance (Gflop/s)	Maximum Utilization (percent)	Size (N×N)	Maximum Performance (Gflop/s)	Maximum Utilization (percent)
LB1	1014	1.644	65.8	1026	1.469	91.8
LB2	1464	1.419	56.3	1026	1.363	85.2
LB3	1538	1.427	57.1	1026	1.321	82.6
FD	801	1.345	53.8	1602	1.228	76.8

4. Conclusions. We have introduced and studied three lattice Boltzmann methods for the 2-D convection–diffusion equation (1). The study included theoretical results such as conditions for convergence, the numerical verification of those results, and a detailed comparison of the methods with an analogous finite difference method. We conclude with the following observations:

1. *Domain of monotonicity.* LB1 and LB2 possess larger domains of monotonicity than does LB3. Further, the domain of monotonicity for LB3 is not preserved from one time step to another; hence, convergence for LB3 was obtained only for those initial conditions not only bounded by the domain of monotonicity but also bounded (much) more severely.
2. *Accuracy.* The accuracy improved progressively from LB3 to LB2 to LB1 over the test cases. LB1 exhibited the greatest accuracy; it was a little less than one (1) to about three (3) times worse than the accuracy exhibited by the finite difference method.
3. *Order of convergence.* Progressively from LB3 to LB2 to LB1, the lattice Boltzmann methods exhibited stronger and stronger evidence for second-order convergence. The evidence for the finite difference method was the strongest, however, with LB1 a close second.
4. *Performance measurements.* Generally, given a particular problem size the level of performance attained increased from the finite difference method to LB3 to LB2 to LB1, with LB1 achieving the highest utilization of the two systems tested.
5. *Timing measurements.* Timing measurements were compared given a particular problem size, i.e., without regard to accuracy. The computations of LB3 took twice the time of the finite difference method, and those of LB1 and LB2 each took about one and a half the time of the finite difference method.
6. *Memory requirements.* The lattice Boltzmann implementations required four times as much memory as the finite difference implementation. (At the expense of increased memory references, the lattice Boltzmann memory usage can be halved by employing "in place" calculations and a separate "advection step.") While in diffusive lattice Boltzmann methods this may be expected, it is not necessarily so in methods for systems involving multiple dependent variables, e.g., lattice Boltzmann methods for the Navier–Stokes equations. For example, the FHP model [9] for the 2-D Navier–Stokes equations requires six values per node, whereas a standard finite difference method would require three or five values per node, depending on whether a Eulerian or Lagrangian formulation is used. The 24-velocity model for the three-dimensional (3-D) Navier–Stokes equations can be reduced to 18 values per node [8]. A standard Navier–Stokes implementation would require four or seven values per node, depending on the formulation. The 2-D and 3-D Navier–Stokes lattice gas methods use at worst two and four and a half times as much memory as the finite difference methods, respectively. Additionally, it is unclear just how much precision is required in the lattice Boltzmann methods in order to obtain a certain level of accuracy. In lattice Boltzmann methods with the exclusion principle, it seems that full precision may not be necessary, depending on the objectives of the calculations. (The computations for the results presented herein were all performed in 64-bit arithmetic.) Perhaps even fixed point arithmetic can be used.
7. *Conservation laws and accuracy.* The computational results suggest that a lattice Boltzmann method attains greater accuracy the more the collision rules are individually faithful to the overriding local conservation laws.
8. *Entropy.* The lattice Boltzmann methods carefully control dissipation. Indeed, their entropy condition implies that they will dissipate physically [7, 10].

Appendices.

A. Analysis of LB1. Following are the details for the collision operator, linearized collision operator, discrete Hilbert expansion, and the consistency conditions.

Simplified, the collision operator can be written as follows:

$$
\begin{aligned}
\mathcal{C}(F)(\mathrm{v}_0) &= \tfrac{1+\epsilon}{4}[F(\mathrm{v}_0)+F(\mathrm{v}_1)+F(\mathrm{v}_2)+F(\mathrm{v}_3)]-F(\mathrm{v}_0)\\
&\quad -\tfrac{\epsilon}{6}[(F(\mathrm{v}_0)+F(\mathrm{v}_2))(F(\mathrm{v}_1)+F(\mathrm{v}_3))+F(\mathrm{v}_0)F(\mathrm{v}_2)+F(\mathrm{v}_1)F(\mathrm{v}_3)],\\
\mathcal{C}(F)(\mathrm{v}_1) &= \tfrac{1}{4}[F(\mathrm{v}_0)+F(\mathrm{v}_1)+F(\mathrm{v}_2)+F(\mathrm{v}_3)]-F(\mathrm{v}_1),\\
\mathcal{C}(F)(\mathrm{v}_2) &= \tfrac{1-\epsilon}{4}[F(\mathrm{v}_0)+F(\mathrm{v}_1)+F(\mathrm{v}_2)+F(\mathrm{v}_3)]-F(\mathrm{v}_2)\\
&\quad +\tfrac{\epsilon}{6}[(F(\mathrm{v}_0)+F(\mathrm{v}_2))(F(\mathrm{v}_1)+F(\mathrm{v}_3))+F(\mathrm{v}_0)F(\mathrm{v}_2)+F(\mathrm{v}_1)F(\mathrm{v}_3)],\\
\mathcal{C}(F)(\mathrm{v}_3) &= \tfrac{1}{4}[F(\mathrm{v}_0)+F(\mathrm{v}_1)+F(\mathrm{v}_2)+F(\mathrm{v}_3)]-F(\mathrm{v}_3)\,.
\end{aligned}
\tag{22}
$$

The linearized collision operator is given by

$$
\mathcal{L}=\tfrac{1}{4}\begin{bmatrix} -3 & 1 & 1 & 1\\ 1 & -3 & 1 & 1\\ 1 & 1 & -3 & 1\\ 1 & 1 & 1 & -3 \end{bmatrix},
$$

which is the circulant $\frac{1}{4}$ Circ$[-3, 1, 1, 1]$, and has the eigenvalues $\boldsymbol{\lambda}(\mathcal{L}) = (\lambda_0, \lambda_1, \lambda_2, \lambda_3) = (0, -1, -1, -1)$. (The corresponding eigenvectors are given in the text.)

The first few terms of the Hilbert expansion (2.3) are given as follows[5]:

1. $g^{(1)} = \sum_{k=0}^{3} c_k^{(1)}\mathbf{q}_k$, where $c_0^{(1)} = 0$, $c_1^{(1)} = -KA(u) + Lu_x$, $c_2^{(1)} = Lu_y$, and $c_3^{(1)} = 0$.
2. $g^{(2)} = \sum_{k=0}^{3} c_k^{(2)}\mathbf{q}_k$, where

$$
\begin{aligned}
c_0^{(2)} &= Tu_t + \tfrac{KL}{2}A'(u)u_x - \tfrac{L^2}{4}\nabla^2 u\,, & c_1^{(2)} &= L\sigma^{(1)}{}_x - KA'(u)\sigma^{(1)}\,,\\
c_2^{(2)} &= L\sigma^{(1)}{}_y\,, & c_3^{(2)} &= \tfrac{KL}{2}A'(u)u_x - \tfrac{L^2}{4}\nabla\cdot\overline{\overline{\nabla}}u\,.
\end{aligned}
$$

3. $g^{(3)} = \sum_{k=0}^{3} c_k^{(3)}\mathbf{q}_k$, where

$$
\begin{aligned}
c_0^{(3)} &= T\sigma^{(1)}{}_t + \tfrac{KL}{2}\partial_x[A'(u)\sigma^{(1)}] - \tfrac{L^2}{4}\nabla^2\sigma^{(1)}\,,\\
c_1^{(3)} &= L\sigma^{(2)}{}_x + K[1-6A'(u)\sigma^{(2)})] + \tfrac{KL^2}{4}A'(u)\nabla^2 u - \tfrac{K^2L}{2}(A'(u))^2u_x\\
&\quad - \tfrac{L^3}{12}(u_{xxx}+3u_{xyy})\,,\\
c_2^{(3)} &= L\sigma^{(2)}{}_y + \tfrac{KL^2}{2}[A'(u)u_{xy}+2u_xu_y] - \tfrac{L^3}{12}(3u_{xxy}+u_{yyy})\,,\\
c_3^{(3)} &= -\tfrac{L^2}{4}\nabla\cdot\overline{\overline{\nabla}}\sigma^{(1)} + \tfrac{KL}{2}\partial_x[A'(u)\sigma^{(1)}]\,.
\end{aligned}
$$

4. $g^{(4)} = \sum_{k=0}^{3} c_k^{(4)}\mathbf{q}_k$, where

$$
\begin{aligned}
c_0^{(4)} &= T\sigma^{(2)}{}_t - \tfrac{L^2}{4}\nabla^2\sigma^{(2)} + 3KL\partial_x[A'(u)\sigma^{(2)}]\\
&\quad + \tfrac{L^4}{96}(u_{xxxx}+6u_{xxyy}+u_{yyyy}) + \tfrac{K^2L^2}{8}A'(u)[A'(u)u_{xx}-4(u_x)^2]\\
&\quad + \tfrac{KL^3}{24}[6(u_xu_{yy}+u_yu_{xy}) - A'(u)(u_{xxx}+3u_{xyy})]\,,\\
c_1^{(4)} &= L\sigma^{(3)}{}_x + \tfrac{K}{2}[4-A'(u)\sigma^{(3)}]\,,\\
c_2^{(4)} &= L\sigma^{(3)}{}_y\,,\\
c_3^{(4)} &= -\tfrac{L^2}{4}\nabla\cdot\overline{\overline{\nabla}}\sigma^{(2)} + 3KL\partial_x[A'(u)\sigma^{(2)}] - \tfrac{L^4}{24}\nabla^3\cdot\overline{\overline{\nabla}}u\\
&\quad + \tfrac{KL^3}{6}[3u_xu_{xx} - A'(u)u_{xxx} - 3u_yu_{xy}] + \tfrac{K^2L^2}{4}A'(u)[A'(u)u_{xx}-4(u_x)^2]\,.
\end{aligned}
$$

[5] Note that lower-order consistency conditions have been used to simplify the coefficients of higher-order coefficients (this includes applying the choice that $\sigma^{(1)} = 0$ where appropriate).

5. $g^{(5)} = \sum_{k=0}^{3} c_k^{(5)}\mathbf{q}_k$, where $c_0^{(5)} = T\sigma^{(3)}{}_t - \frac{L^2}{4}\nabla^2\sigma^{(3)} + \frac{KL}{4}\partial_x[A'(u)\sigma^{(3)}]$, and the remaining coefficients $c_1^{(5)}$, $c_2^{(5)}$, and $c_3^{(5)}$ are not given because they are not used in the proofs.

The following are the consistency conditions for the first few orders of the Hilbert expansion:

1. $\mathcal{O}\big[\delta^2\big]$: $\partial_t u + c\partial_x A(u) = \nu\nabla^2 u$, in which $c = \frac{KL}{2T}$, $\nu = \frac{L^2}{4T}$, and $A(u) = u(1-u)$.
2. $\mathcal{O}\big[\delta^3\big]$: $\partial_t\sigma^{(1)} + c\partial_x[A'(u)\sigma^{(1)}] = \nu\nabla^2\sigma^{(1)}$.
3. $\mathcal{O}\big[\delta^4\big]$: $\partial_t\sigma^{(2)} + 6c\partial_x[A'(u)\sigma^{(2)}] = \nu\nabla^2\sigma^{(2)} - \mathcal{F}$, where $\mathcal{F} = \frac{\nu}{24}[u_{xxxx} + 6u_{xxyy} + u_{yyyy}] - \frac{KL^3}{24T}[A'(u)(u_{xxx}+3u_{xyy}) - 6(u_x u_{yy} + u_y u_{xy}) - \frac{K^2L^2}{8T}A'(u)[4(u_x)^2 - A'(u)u_{xx}]$.
4. $\mathcal{O}\big[\delta^5\big]$: $\partial_t\sigma^{(3)} + \frac{c}{2}\partial_x[A'(u)\sigma^{(3)}] = \nu\nabla^2\sigma^{(3)}$, which emits the exact solution $\sigma^{(3)} = 0$.

B. Analysis of LB2. The collision operator can be written as follows:

$$
\begin{aligned}
\mathcal{C}(F)(\mathrm{v}_0) &= \tfrac{1+\epsilon}{4}[F(\mathrm{v}_0) + F(\mathrm{v}_1) + F(\mathrm{v}_2) + F(\mathrm{v}_3)] - F(\mathrm{v}_0)\\
&\quad - \tfrac{\epsilon}{2}[F(\mathrm{v}_0)F(\mathrm{v}_2) + F(\mathrm{v}_1)F(\mathrm{v}_3)],\\
\mathcal{C}(F)(\mathrm{v}_1) &= \tfrac{1}{4}[F(\mathrm{v}_0) + F(\mathrm{v}_1) + F(\mathrm{v}_2) + F(\mathrm{v}_3)] - F(\mathrm{v}_1),\\
\mathcal{C}(F)(\mathrm{v}_2) &= \tfrac{1-\epsilon}{4}[F(\mathrm{v}_0) + F(\mathrm{v}_1) + F(\mathrm{v}_2) + F(\mathrm{v}_3)] - F(\mathrm{v}_2)\\
&\quad + \tfrac{\epsilon}{2}[F(\mathrm{v}_0)F(\mathrm{v}_2) + F(\mathrm{v}_1)F(\mathrm{v}_3)],\\
\mathcal{C}(F)(\mathrm{v}_3) &= \tfrac{1}{4}[F(\mathrm{v}_0) + F(\mathrm{v}_1) + F(\mathrm{v}_2) + F(\mathrm{v}_3)] - F(\mathrm{v}_3)\,.
\end{aligned}
\tag{23}
$$

The linearized collision operator $\mathcal{L}$ is as in LB1 and consequently so are its eigenvalues and eigenvectors.

The first few terms of the Hilbert expansion[5] are given as follows:

1. $g^{(1)} = \sum_{k=0}^{3} c_k^{(1)}\mathbf{q}_k$, where $c_0^{(1)} = c_3^{(1)} = 0$, $c_1^{(1)} = -KA(u) + Lu_x$, and $c_2^{(1)} = Lu_y$.
2. $g^{(2)} = \sum_{k=0}^{3} c_k^{(2)}\mathbf{q}_k$, where

$$
\begin{aligned}
c_0^{(2)} &= Tu_t + \tfrac{KL}{2}A'(u)u_x - \tfrac{L^2}{4}\nabla^2 u\,, & c_1^{(2)} &= L\sigma^{(1)}{}_x - KA'(u)\sigma^{(1)}\,,\\
c_2^{(2)} &= L\sigma^{(1)}{}_y\,, & c_3^{(2)} &= \tfrac{KL}{2}A'(u)u_x - \tfrac{L^2}{4}\nabla\cdot\overline{\nabla}u\,.
\end{aligned}
$$

3. $g^{(3)} = \sum_{k=0}^{3} c_k^{(3)}\mathbf{q}_k$, where

$$
\begin{aligned}
c_0^{(3)} &= T\sigma^{(1)}{}_t + \tfrac{KL}{2}\partial_x[A'(u)\sigma^{(1)}] - \tfrac{L^2}{4}\nabla^2\sigma^{(1)}\,,\\
c_1^{(3)} &= L\sigma^{(2)}{}_x + K[1 - 6A'(u)\sigma^{(2)})] + \tfrac{KL^2}{4}A'(u)\nabla^2 u - \tfrac{K^2L}{2}(A'(u))^2 u_x\\
&\quad - \tfrac{L^3}{12}(u_{xxx} + 3u_{xyy})\,,\\
c_2^{(3)} &= L\sigma^{(2)}{}_y - \tfrac{L^3}{12}(u_{xxx} + 3u_{xyy}) + \tfrac{KL^2}{2}(A'(u)u_y)\,,\\
c_3^{(3)} &= -\tfrac{L^2}{4}\nabla\cdot\overline{\nabla}\sigma^{(1)} + \tfrac{KL}{2}\partial_x[A'(u)\sigma^{(1)}]\,.
\end{aligned}
$$

4. $g^{(4)} = \sum_{k=0}^{3} c_k^{(4)}\mathbf{q}_k$, where

$$
\begin{aligned}
c_0^{(4)} &= T\sigma^{(2)}{}_t - \tfrac{L^2}{4}\nabla^2\sigma^{(2)} + 3KL\partial_x[A'(u)\sigma^{(2)}] + \tfrac{L^4}{96}(u_{xxxx} + 6u_{xxyy} + u_{yyyy})\\
&\quad + \tfrac{KL^3}{24}[6(u_x u_{yy} + u_y u_{xy}) - A'(u)(u_{xxx} + 3u_{xyy})]\\
&\quad + \tfrac{K^2L^2}{8}A'(u)[u_{xx} - 4(u_x)^2]\,,\\
c_1^{(4)} &= L\sigma^{(3)}{}_x + \tfrac{K}{2}[4 - A'(u)\sigma^{(3)}]\,,\\
c_2^{(4)} &= L\sigma^{(3)}{}_y\,,\\
c_3^{(4)} &= -\tfrac{L^2}{4}\nabla\cdot\overline{\nabla}\sigma^{(2)} + 3KL\partial_x[A'(u)\sigma^{(2)}] + \tfrac{L^4}{24}\nabla^3\cdot\overline{\nabla}u\\
&\quad + \tfrac{KL^3}{6}[3u_x u_{xx} - A'(u)u_{xxx} - 3u_y u_{xy}] + \tfrac{K^2L^2}{4}A'(u)[A'(u)u_{xx} - 4(u_x)^2]\,.
\end{aligned}
$$

5. $g^{(5)} = \sum_{k=0}^{3} c_k^{(5)} \mathbf{q}_k$, where $c_0^{(5)} = T\sigma^{(3)}{}_t - \frac{L^2}{4}\nabla^2\sigma^{(3)} + \frac{KL}{4}\partial_x[A'(u)\sigma^{(3)}]$, and the remaining coefficients $c_1^{(5)}$, $c_2^{(5)}$, and $c_3^{(5)}$ are not given because they are not used in the proofs.

The following are the consistency conditions for the first few orders of the Hilbert expansion:

1. $\mathcal{O}\big[\delta^2\big]$: $\partial_t u + c\partial_x A(u) = \nu\nabla^2 u$, in which $c = \frac{KL}{2T}$, $\nu = \frac{L^2}{4T}$, and $A(u) = u(1-u)$.
2. $\mathcal{O}\big[\delta^3\big]$: $\partial_t\sigma^{(1)} + c\partial_x[A'(u)\sigma^{(1)}] = \nu\nabla^2\sigma^{(1)}$.
3. $\mathcal{O}\big[\delta^4\big]$: $\partial_t\sigma^{(2)} + 6c\partial_x[A'(u)\sigma^{(2)}] = \nu\nabla^2\sigma^{(2)} - \mathcal{F}$, where $\mathcal{F} = \frac{\nu}{24}[u_{xxxx} + 6u_{xxyy} + u_{yyyy}] - \frac{KL^3}{24T}[A'(u)(u_{xxx} + 3u_{xyy}) - 6(u_x u_{yy} + u_y u_{xy})] - \frac{K^2L^2}{8T}A'(u)[4(u_x)^2 - A'(u)u_{xx}]$.
4. $\mathcal{O}\big[\delta^5\big]$: $\partial_t\sigma^{(3)} + \frac{c}{2}\partial_x[A'(u)\sigma^{(3)}] = \nu\nabla^2\sigma^{(3)}$, which emits the exact solution $\sigma^{(3)} = 0$.

C. Analysis of LB3. The collision operator can be written as follows:

$$
\begin{aligned}
\mathcal{C}(F)(\mathrm{v}_0) &= \tfrac{1+\epsilon}{2}[F(\mathrm{v}_0) + F(\mathrm{v}_2)] - F(\mathrm{v}_0) \\
&\quad + F(\mathrm{v}_0)F(\mathrm{v}_2)[F(\mathrm{v}_1) + F(\mathrm{v}_3) - (1+\epsilon)] - F(\mathrm{v}_1)F(\mathrm{v}_3)[F(\mathrm{v}_0) + F(\mathrm{v}_2) - 1], \\
\mathcal{C}(F)(\mathrm{v}_1) &= \tfrac{1}{2}[F(\mathrm{v}_1) + F(\mathrm{v}_3)] - F(\mathrm{v}_1) \\
&\quad - F(\mathrm{v}_0)F(\mathrm{v}_2)(F(\mathrm{v}_1) + F(\mathrm{v}_3) - 1) + F(\mathrm{v}_1)F(\mathrm{v}_3)(F(\mathrm{v}_0) + F(\mathrm{v}_2) - 1), \\
\mathcal{C}(F)(\mathrm{v}_2) &= \tfrac{1-\epsilon}{2}[F(\mathrm{v}_0) + F(\mathrm{v}_2)] - F(\mathrm{v}_2) \\
&\quad + F(\mathrm{v}_0)F(\mathrm{v}_2)[F(\mathrm{v}_1) + F(\mathrm{v}_3) - (1-\epsilon)] - F(\mathrm{v}_1)F(\mathrm{v}_3)[F(\mathrm{v}_0) + F(\mathrm{v}_2) - 1], \\
\mathcal{C}(F)(\mathrm{v}_3) &= \tfrac{1}{2}[F(\mathrm{v}_1) + F(\mathrm{v}_3)] - F(\mathrm{v}_3) \\
&\quad - F(\mathrm{v}_0)F(\mathrm{v}_2)[F(\mathrm{v}_1) + F(\mathrm{v}_3) - 1] + F(\mathrm{v}_1)F(\mathrm{v}_3)[F(\mathrm{v}_0) + F(\mathrm{v}_2) - 1].
\end{aligned}
\tag{24}
$$

The linearized collision operator is given by $\mathcal{L} = \frac{1}{2}\,\mathrm{Circ}[-1 - 2u + 2u^2, 2A(u), 1 - 2u + 2u^2, 2A(u)]$, which has the eigenvalues $\boldsymbol{\lambda}(\mathcal{L}) = (\lambda_0, \lambda_1, \lambda_2, \lambda_3) = (0, -1, -1, -4A(u))$. (The corresponding eigenvectors are given in the main text.)

The first few terms of the Hilbert expansion (2.3) for LB3 are similar to the expansions for LB1 and LB2; however, the terms are somewhat more complicated. Let $\lambda \equiv \lambda_3$. Then the first three terms of the Hilbert expansion for LB3 follow[5]:

1. $g^{(1)} = \sum_{k=0}^{3} c_k^{(1)} \mathbf{q}_k$, where $c_0^{(1)} = c_3^{(1)} = 0$, $c_1^{(1)} = -KA(u) + Lu_x$, and $c_2^{(1)} = Lu_y$.
2. $g^{(2)} = \sum_{k=0}^{3} c_k^{(2)} \mathbf{q}_k$, where

$$
\begin{aligned}
c_0^{(2)} &= Tu_t + \tfrac{KL}{2}A'(u)u_x - \tfrac{L^2}{4}\nabla^2 u, \qquad c_1^{(2)} = L\sigma^{(1)}{}_x - KA'(u)\sigma^{(1)}, \\
c_2^{(2)} &= L\sigma^{(1)}{}_y, \\
c_3^{(2)} &= \tfrac{L^2}{4}(4uu_x^2 - \overline{\nabla}u) + \tfrac{KL}{2}u_x[A'(u) - 4uA(u)] + K^2u[A(u)]^2 \\
&\quad + u[\sigma^{(1)}(\sigma^{(1)} - 2Lu_y) - 1].
\end{aligned}
$$

3. $g^{(3)} = \sum_{k=0}^{3} c_k^{(3)} \mathbf{q}_k$, where

$$
\begin{aligned}
c_0^{(3)} &= T\sigma^{(1)}{}_t + \tfrac{KL}{2}\partial_x[A'(u)\sigma^{(1)}] - \tfrac{L^2}{4}\nabla^2\sigma^{(1)}, \\
c_1^{(3)} &= K(1 - 6\sigma^{(2)} + 12u\sigma^{(2)}) + L\sigma^{(2)}{}_x - \tfrac{L^3}{12\lambda}[u_{xxx}(3 + 4\lambda) - 3u_{xyy}] \\
&+ \tfrac{L^3}{\lambda}u_x(uu_{xx} + u_x^2) + \tfrac{L^3}{4\lambda^2}\lambda_x(u_{xx} - 4uu_x^2 - u_{yy}) \\
&- \tfrac{2L^2}{\lambda}(\sigma^{(1)}uu_{xy} + uu_y\sigma^{(1)}{}_x + u_xu_y\sigma^{(1)}) + \tfrac{2L^2}{\lambda^2}\lambda_x uu_y\sigma^{(1)} \\
&+ \tfrac{L}{\lambda}[2u\sigma^{(1)}\sigma^{(1)}{}_x - u_x(1 - (\sigma^{(1)})^2)] + \tfrac{L}{\lambda^2}\lambda_x u[1 - (\sigma^{(1)})^2] - KL^2{u_x}^2
\end{aligned}
$$

$$\begin{aligned}
&+ \tfrac{KL^2}{4} A'(u)[3u_{xx} + u_{yy}] - \tfrac{KL^2}{\lambda} u_x^2 (1 + 10u - 18u^2) \\
&+ \tfrac{2KL^2}{\lambda} u_{xx}[A'(u) - uA(u)] - \tfrac{3KL^2}{2\lambda} u_{yy} A'(u) \\
&- \tfrac{KL^2}{2\lambda^2} \lambda_x u_x [A'(u) - 4uA(u)] + \tfrac{6K}{\lambda} uA'(u)[1 - (\sigma^{(1)})^2] \\
&+ \tfrac{12KL}{\lambda} uu_y \sigma^{(1)} A'(u) - \tfrac{K^2L}{2} u_x [A'(u)]^2 \\
&- \tfrac{K^2L}{\lambda} u_x (3 - 12u - 3u^2 + 44u^3 - 29u^4) \\
&- \tfrac{K^2L}{\lambda^2} u\lambda_x [A(u)]^2 - \tfrac{6K^3}{\lambda} uA'(u)[A(u)]^2 , \\
c_2^{(3)} &= L\sigma^{(2)}{}_y - \tfrac{L}{\lambda^2} u\lambda_y [1 - (\sigma^{(1)})^2] - \tfrac{L}{\lambda}[2u\sigma^{(1)}{}_y \sigma^{(1)} - u_y(1 - (\sigma^{(1)})^2)] \\
&- \tfrac{L^3}{3} u_{yyy} + \tfrac{L^3}{4\lambda}[\partial_y \overline{\overline{\nabla}} u - 4u_x(u_x u_y + 2uu_{xy})] - \tfrac{L^3}{4\lambda^2} \lambda_y [\overline{\overline{\nabla}} u - 4uu_x^2] \\
&+ \tfrac{K^2L}{\lambda^2} \lambda_y u [A(u)]^2 + \tfrac{2L^2}{\lambda^2}[\lambda(uu_y \sigma^{(1)}{}_y + u_y^2 \sigma^{(1)} + u\sigma^{(1)} u_{yy}) - u\sigma^{(1)} \lambda_y] \\
&+ \tfrac{KL^2}{2\lambda^2} \lambda_y u_x [A'(u) - 4uA(u)] + \tfrac{KL^2}{\lambda} u_x u_y (1 + 4u - 6u^2) \\
&- \tfrac{KL^2}{2\lambda} u_{xy}[A'(u) - 4uA(u)] - \tfrac{K^2L}{\lambda} u_y u A(u)(3 - 5u) , \\
c_3^{(3)} &= -2u(1 - \sigma^{(1)}\sigma^{(2)}) + \tfrac{2}{\lambda} u^2 \sigma^{(1)}[1 - (\sigma^{(1)})^2] + \tfrac{L^3}{2\lambda} uu_y [\overline{\overline{\nabla}} u - 4uu_x^2] \\
&+ \tfrac{2L}{\lambda} u^2 u_y [1 + (\sigma^{(1)})^2] - 2Lu(\sigma^{(1)}\sigma^{(1)}{}_y + u_y + \sigma^{(2)}) \\
&- \tfrac{L^2}{4}[\overline{\overline{\nabla}} u - 8uu_x \sigma^{(1)}{}_x] + \tfrac{L^2}{2\lambda} u\sigma^{(1)}[\overline{\overline{\nabla}} u - 4u(u_x^2 - 2u_y^2)] \\
&- \tfrac{KL^2}{\lambda} uu_x u_y [A'(u) - 4uA(u)] - \tfrac{2K^2L}{\lambda} u^2 u_y [A(u)]^2 \\
&+ \tfrac{KL}{2} \sigma^{(1)}{}_x [A'(u) - 4uA(u)] - KLu_x \sigma^{(1)}(1 + 2u - 4u^2) \\
&- \tfrac{KL}{\lambda} uu_x \sigma^{(1)}[A'(u) - 4uA(u)] + 2K^2 u\sigma^{(1)} A(u) A'(u) \\
&- \tfrac{2K^2}{\lambda} u^2 \sigma^{(1)} [A(u)]^2 .
\end{aligned}$$

4. $g^{(4)} = \sum_{k=0}^{3} c_k^{(4)} \mathbf{q}_k$, where all the coefficients are nonzero—even with $\sigma^{(1)} = 0$. We do not give them here since they are much too lengthy for the present text. Note, however, that $c_0^{(4)}$ can be easily determined from the $\mathcal{O}\big[\delta^4\big]$ consistency condition, which is given below.

Let $c = \frac{KL}{2T}$, $\nu = \frac{L^2}{4T}$, and $A(u) = u(1-u)$. Then the first few consistency conditions are as follows:

1. $\mathcal{O}\big[\delta^2\big]$: $\partial_t u + c\partial_x A(u) = \nu\nabla^2 u$.
2. $\mathcal{O}\big[\delta^3\big]$: $\partial_t \sigma^{(1)} + c\partial_x[A'(u)\sigma^{(1)}] = \nu\nabla^2\sigma^{(1)}$.
3. $\mathcal{O}\big[\delta^4\big]$: $\partial_t \sigma^{(2)} + 6c\partial_x[A'(u)\sigma^{(2)}] = \nu\nabla^2\sigma^{(2)} - \mathcal{F}$, where

$$\begin{aligned}
\mathcal{F} = &\tfrac{\nu}{2\lambda^3}\big[((\lambda_x)^2 - (\lambda_y)^2)(u_{xx} + 4u(u_x)^2 - u_{yy})\big] \\
&- \tfrac{\nu}{4\lambda^2}[2(\lambda_x \nabla\cdot\overline{\overline{\nabla}} u_x - \lambda_y \nabla\cdot\overline{\overline{\nabla}} u_y) + (\lambda_{xx}\nabla\cdot\overline{\overline{\nabla}} u - \lambda_{yy}\nabla\cdot\overline{\overline{\nabla}} u) \\
&\quad - 16uu_x(\lambda_x u_{xx} + \lambda_y u_{xy}) - 4u(u_x)^2 \nabla\cdot\overline{\overline{\nabla}}\lambda + 8(u_x)^2 u_y \lambda_y \\
&\quad - 16uu_x(\lambda_x u_{xx} + \lambda_y u_{xy}) - 8(u_x)^3 \lambda_x] + \tfrac{\nu}{4\lambda}[(u_{xxxx} + 2u_{xxyy} - u_{yyyy}) \\
&\quad - 8uu_x(u_{xxx} - u_{yyy}) - 8u(u_{xx})^2 - (u_x)^2(20u_{xx} - 4u_{yy}) \\
&\quad + 8(2u_x u_y + uu_{xy})u_{xy}] + \tfrac{\nu}{24}[7u_{xxxx} - 6u_{xxyy} + 7u_{yyyy}] \\
&- \tfrac{KL^3}{4T\lambda^3}(1 - 2u - 4u^2 + 4u^3)[(\lambda_x)^2 u_x - (\lambda_y)^2 u_y] \\
&+ \tfrac{KL^3}{8T\lambda^3}\big[8(1 - 2u - u^2 - u^3)\lambda_x u_{xx} - 6(1-u)\lambda_x u_{yy} \\
&\quad + (1 - 2u - 4u^2 + 4u^3)[(u_x \nabla\cdot\overline{\overline{\nabla}}\lambda) - \lambda_y u_{xy}] \\
&\quad - 4(1 + 10u - 18u^2)\lambda_x(u_x)^2 + 4(1 + 4u - 6u^2)\lambda_y u_x u_y\big] \\
&- \tfrac{KL^3}{8T\lambda}[(7 - 14u - 4u^2 - 4u^3)\nabla\cdot\overline{\overline{\nabla}} u_x - (18 + 72u - 132u^2)u_x u_{xx}
\end{aligned}$$

$$
\begin{aligned}
&- \; 8(4-14u)(u_x)^3 + 2(7+4u-6u^2)u_x u_{yy} + 2(2+41u-12u^2)u_y u_{xy} \\
&+ \; 8(1-3u)u_x(u_y)^2] - \tfrac{KL^3}{12T}[2A'(u)u_{xxx} - 9u_x u_{xx} + 3u_y u_{xy}] \\
+ \; & \tfrac{2\nu}{\lambda^3}u[(\lambda_x)^2 - (\lambda_y)^2] - \tfrac{\nu}{\lambda^2}[(u\nabla\cdot\overline{\nabla}\lambda) + 2(\lambda_x u_x - \lambda_y u_y)] + \tfrac{\nu}{\lambda}\nabla\cdot\overline{\nabla}u \\
- \; & \tfrac{K^2L^2}{2T\lambda^3}u^3(1-2u+u^2)[(\lambda_x)^2-(\lambda_y)^2] + \tfrac{K^2L^2}{4T\lambda^2}[u^3(1-2u+u^2)(\lambda_{xx}-\lambda_{yy}) \\
&- \; (7-4u-6u^2+88u^3-58u^4)\lambda_x u_x - 2u^2(3-8u+5u^2)\lambda_y u_y] \\
+ \; & \tfrac{K^2L^2}{4T\lambda}[(6-24u-3u^2+80u^3-212u^4)u_{xx} + u^2(3-8u+5u^2)u_{yy} \\
&- \; (24+6u+240u^2-212u^3)(u_x)^2 + 2u(3-4u+u^2)(u_y)^2] \\
- \; & \tfrac{K^2L^2}{8T}[4A'(u)(u_x)^2 - (A'(u))^2u_{xx}] - \tfrac{3K^3L}{T\lambda^2}u^3(1-4u+5u^2-2u^3) \\
- \; & \tfrac{3KL}{T\lambda}(1-4u)u_x + \tfrac{3KL}{T\lambda^2}uA'(u)\lambda_x + \tfrac{3K^3L}{T\lambda}u^2(3-16u+25u^2-12u^3).
\end{aligned}
$$

4. $\mathcal{O}[\delta^5]$:

$$\partial_t\sigma^{(3)} + \tfrac{c}{2}\partial_x[A'(u)\sigma^{(3)}] = \nu\nabla^2\sigma^{(3)} - \mathcal{G}\,, \tag{25}$$

where for $u \in C^4((0,1)^2,(0,1))$, $\mathcal{G}$ is a smooth function of the constants T, K, and L, and u and $\sigma^{(2)}$ and their derivatives. In general, $\mathcal{G}$ is nonzero, and hence, contrary to the cases for LB1 and LB2, $\sigma^{(3)} = 0$ is not a solution of the fifth-order consistency condition (25). The fact the $\sigma^{(3)} = 0$ is *not* a solution to (25) may attest to the unfavorable numerical results for LB3.

Acknowledgments. The author thanks Professors C. D. Levermore (University of Arizona) and G. H. Rodrigue (University of California, Davis and LLNL), Dr. S. E. Zarantonello (Silicon Graphics Computer Systems), Messrs. S. Gavali (Fujitsu America, Inc. (FAI)) and T. Sands (FAI), and Ms. L. Tingley (FAI) for their valued comments and suggestions in reviewing the manuscript; Professor H. A. van der Vorst (University of Utrecht, The Netherlands) for an insightful and enjoyable discussion at Supercomputing '92 (Minneapolis, MN); and Dr. S. Succi (IBM-ECSEC, Rome, Italy) for valuable discussions.

REFERENCES

[1] N. Bellomo, A. Palczewski, and G. Toscani, *Mathematical Topics in Nonlinear Kinetic Theory*, World Scientific, River Edge, NJ, 1988.

[2] B. M. Boghosian and W. Taylor, *Correlations and renormalization in lattice gases*, Phys. Rev. Lett. E, 52 (1995), p. 510.

[3] G. D. Doolen, *Lattice gas methods: Theory, applications and hardware*, Physica D, 47 (1991), pp. 299–337.

[4] A. B. H. Elton, *A Numerical Theory of Lattice Gas and Lattice Boltzmann Methods in the Computation of Solutions to Nonlinear Advective-Diffusive Systems*, Ph.D. thesis, University of California, Davis, CA, September 1990. Available as Lawrence Livermore National Laboratory Tech. Report #UCRL–LR–105090.

[5] B. H. Elton, *A lattice Boltzmann method for a two-dimensional viscous Burgers equation: Computational results*, in Proceedings Supercomputing '91, Los Alamitos, CA, 1991, IEEE Computer Society Press, Piscataway, NJ, pp. 242–252.

[6] B. H. Elton, C. D. Levermore, and G. H. Rodrigue, *Lattice Boltzmann methods for some 2-D nonlinear diffusion equations: Computational results*, in Asymptotic Analysis and Numerical Solution of Partial Differential Equations, Lecture Notes in Pure and Applied Mathematics 130, Marcel Dekker Inc., New York, 1990, pp. 197–214.

[7] ———, *Convergence of convective-diffusive lattice Boltzmann methods*, SIAM J. Numer. Anal., 32 (1995), pp. 1327–1354.

[8] U. Frisch, D. D'Humières, B. Hasslacher, P. Lallemand, Y. Pomeau, and J.-P. Rivet, *Lattice gas hydrodynamics in two and three dimensions*, Complex Systems, 1 (1987), pp. 649–707.

[9] U. Frisch, B. Hasslacher, and Y. Pomeau, *Lattice gas automata for the Navier–Stokes equation*, Phys. Rev. Lett., 56 (1986), pp. 1505–1508.

[10] D. Levermore, *Fluid dynamical limits of discrete kinetic theories*, in Macroscopic Simulations of Complex Phenomena, M. Mareschal and B. Holian, eds., NATO ASI series A, Plenum Press, New York, 1992, pp. 173–185.

[11] R. D. Richtmyer and K. W. Morton, *Difference Methods for Initial-Value Problems*, Interscience Tracts in Pure and Applied Mathematics, Vol. 4, 2nd ed., Wiley Interscience, New York, 1967, p. 262, p. 323.

[12] G. A. Sod, *Numerical Methods in Fluid Dynamics: Initial and Initial Boundary-Value Problems*, Cambridge University Press, Cambridge, 1988.

[13] N. Uchida, *Fujitsu* VP2000 *series supercomputers*, International Journal of High Speed Computing, 3 (1991), pp. 169–185.

[14] N. Uchida, M. Hirai, M. Yoshida, and K. Hotta, *Fujitsu* VP2000 *series*, in Spring COMPCON '90, IEEE Computer Society Press, Piscataway, NJ, 1990, pp. 4–11.

[15] T. Utsumi, M. Ikeda, and M. Takamura, *Architecture of the* VPP500 *parallel supercomputer*, in Proceedings Supercomputing '94, Los Alamitos, CA, 1994, IEEE Computer Society Press, Piscataway, NJ, pp. 478–487.

SIAM J. Sci. Comput.
Vol. 17, No. 4, pp. 814–829, July 1996

DIRECT NUMERICAL CALCULATIONS OF A NEUTRAL STABILITY CURVE FOR ONE-DIMENSIONAL DETONATIONS*

WEI CAI[†], WONHO OH[‡], AND YOULAN ZHU[§]

Abstract. In this paper we calculate the neutral stability curve for one-dimensional detonation waves based on direct numerical simulations of the Euler equations. We propose multidimensional spectral methods with accurate shock tracking to study the stability of steady Chapman–Jouguet detonations. The numerical methods proposed here can be easily generalized for the simulation of multidimensional detonation waves. The detonations under consideration are for an ideal gas in an irreversible, unimolecular reaction $A \rightarrow B$ with finite Arrhenius reaction rate. Critical overdrives f^* for various heat releases Q's are computed. The neutral stability curve is obtained and compared with the results of the eigenmode analysis by Erpenbeck [*Phys. Fluids*, 7 (1964), pp. 684–696].

Key words. detonation waves, neutral stability, multidomain spectral methods, shock tracking

AMS subject classifications. 76Q05, 76V05

1. Introduction.

The stability of detonation waves is a complex yet fascinating multidimensional phenomenon. There has been renewed interest in understanding the underlying physics of this phenomenon [1, 2, 4, 10, 14, 16]. Most of the observed detonation waves travelling in rectangular or circular channels demonstrate transverse wave structures along the precursor shock front in the form of mach-stems—so-called "triple points." Such structures are believed to play important roles in the stability of multidimensional detonation waves. The study of stability of one-dimensional detonation waves provides a stepping stone to the more complete multidimensional stability problem.

Basically, there have been two approaches in studying the stability of one-dimensional detonations. The first is to study the hydrodynamic stability of the linearized Euler equations with respect to a given steady solution by using eigenmode analysis; the second approach is to use direct numerical simulation of one-dimensional Euler equations. Erpenbeck started the first approach for an ideal gas in an irreversible, unimolecular reaction $A \rightarrow B$ with an Arrhenius reaction rate [8]. Later, Abouseif and Toong [1] derived an approximate wave equation for the perturbed state variables. They successfully identified the sources of longitudinal instability under the assumption of small perturbation and high activation energy.

However, as common to any linearized analysis of a strongly nonlinear problem, it could not explain all the nonlinear phenomena of reacting flows. On the other hand, direct numerical study of one-dimensional detonations includes the work of Fickett and Wood [9] and Fickett, Jacobson, and Schott [10] using a method of characteristics. In [10], they considered two reaction rates with induction zones. Recently, Bourlioux, Majda, and Roytburd [4] have studied the same problem posed by Erpenbeck [8] using both asymptotic eigenmode analysis and improved numerical simulations based on the shock capturing scheme—piecewise parabolic methods (PPM). They provide a detailed numerical convergence study and an asymptotic analysis for the case when the heat release $Q = 50$ in the one step reaction $A \rightarrow B$ and the activation energy $E^+ = 50$ for an ideal gas with gas constant $\gamma = 1.2$.

So far there has been no attempt to calculate the neutral stability curve for all heat release and activation energy parameters based on direct numerical simulations of full Euler

*Received by the editors September 24, 1992; accepted for publication (in revised form) March 1, 1995.

[†] Department of Mathematics, University of California, Santa Barbara, CA 93106 (wcai@newton.math.ucsb.edu). Supported by NSF grants ASC-9005874 and ASC-9113895 and a supercomputing grant from the North Carolina Supercomputing Center.

[‡] Department of Applied Mathematics and Statistics, State University of New York at Stony Brook, Stony Brook, NY 11794-3600 (woh@ams.sunysb.edu).

[§] Department of Mathematics, University of North Carolina at Charlotte, Charlotte, NC 28223 (yzhu@mosaic.uncc.edu).

equations. In Erpenbeck's early work [8], a crude curve of such kind was given based on the eigenmode analysis. In this paper, we will propose a numerical method for computing reacting shock waves and calculate the neutral stability curve for one-dimensional detonations under one-dimensional longitudinal disturbances. Theoretically, similar neutral curves could be sought for the case when the disturbance is two or three dimensional. We believe that such a question of multidimensional stability problem is best answered when simulations are done for multidimensional flows. In this paper, we will concentrate on the one-dimensional stability studies only.

The purpose of this work is twofold. The first is to introduce a high-order numerical method which is designed to fit the simulations of multidimensional detonations; the second is to provide a parametric study of the neutral stability curve for one-dimensional detonations which have aided our study of multidimensional detonations [3]. The novel feature of our algorithm is the high-order scheme used in tracking the detonation front compared with the first-order scheme used in [4]; also a multidomain approach is used so different numerical discretizations can be used in various parts of the flow field, a feature which proves to be important in the calculation of two-dimensional detonation waves where triple shocks occur along the detonation front [3].

2. Stability of steady detonations. We will consider the ideal Zeldovich, von Neumann, and Doering model of a steady detonation that is assumed to have a steady solution with constant shock speed D. According to the classical Chapman–Jouguet theory, there is a minimum shock speed D_j below which no strong detonation is possible. Within a gas dynamic shock transition, there are not enough molecular collisions to cause chemical reactions. Therefore, we will assume no chemical reactions across the shock and will treat the front as a mathematical discontinuity. This will enable us to use a shock tracking algorithm for the shock front.

We adopt the simple model suggested by Erpenbeck in [8] which assumes a one-step $A \rightarrow B$, irreversible exothermic reaction with a finite Arrhenius reaction rate. Here A denotes the unreacted fuel and B the product. Gas particles upon passing across the shock front will initiate chemical reactions immediately according to the Arrhenius rate. The reacting gas for such a system will be governed by the following Euler equations, coupled with an equation for the production of species (only one such equation is needed here, as the other species is obtained from the continuity equation):

$$\frac{\partial \vec{\mathbf{u}}}{\partial t} + \frac{\partial \mathbf{F}(\vec{\mathbf{u}})}{\partial x} = \Psi(\vec{\mathbf{u}}), \tag{1}$$

$$\vec{\mathbf{u}} = \begin{pmatrix} \rho \\ \rho u \\ \rho e \\ \rho\lambda \end{pmatrix}, \qquad \mathbf{F}(\vec{\mathbf{u}}) = \begin{pmatrix} \rho u \\ \rho u + p \\ \rho u e + u p \\ \rho u \lambda \end{pmatrix}, \tag{2}$$

and

$$\Psi(\vec{\mathbf{u}}) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -\omega \end{pmatrix}, \tag{3}$$

where $p = \rho T$, $e = \frac{1}{\gamma-1}\frac{p}{\rho} + \frac{u^2}{2} + \lambda Q$, and $\omega = K\rho\lambda \exp(-\frac{E^+}{T})$. We take the ratio of specific heats $\gamma = 1.2$ in all our computations. ρ, u, p, λ, e, T, E^+, and Q are density, flow velocity, pressure, mass fraction of reactant, total specific energy, temperature, activation energy, and heat release, respectively. Equation (1) also can be rewritten in the nonconservative form

$$\frac{\partial \vec{\mathbf{u}}}{\partial t} + \mathbf{A}\frac{\partial \vec{\mathbf{u}}}{\partial x} = \Psi(\vec{\mathbf{u}}), \tag{4}$$

where $\mathbf{A} = \frac{\partial \mathbf{F}}{\partial \vec{\mathbf{u}}}$ is the Jacobian matrix of flux function $\mathbf{F}(\vec{\mathbf{u}})$.

All quantities in (1) and (4) have been nondimensionalized by their reference quantities in the unreacted region. They are given as follows ("$\longleftarrow$" indicates nondimensionalization and the subscript "0" denotes unreacted gas states):

$$\begin{aligned} p &\longleftarrow \frac{p}{p_0}, \\ \rho &\longleftarrow \frac{\rho}{\rho_0}, \\ u &\longleftarrow \frac{u}{\sqrt{RT_0}}, \\ T &\longleftarrow \frac{T}{T_0}, \\ E^+ &\longleftarrow \frac{E^+}{RT_0}, \\ Q &\longleftarrow \frac{Q}{RT_0}, \\ t &\longleftarrow \frac{t}{t^*}, \\ x &\longleftarrow \frac{x}{\sqrt{RT_0}t^*}, \end{aligned}$$

where t^* is the half reaction time.

We consider an overdriven detonation travelling in a one-dimensional tube. The flow is driven by a piston and the shock front propagating to the right of the tube into unreacted gas. Given the speed of the piston and the initial states of unreacted gas in front of the shock, the flow field can be obtained as the solution of (1) or (4). For the case of overdriven detonation considered here, a steady solution can be easily obtained by solving the Rankine–Hugoniot condition which connects the states in front of the shock to all the states after the shock [11]. The shock speed D parameterizes the family of steady solutions if we assume that the states behind the final state (where chemical reaction terminates) remain constant. The overdrive f is defined as

$$f = \left(\frac{D}{D_j}\right)^2.$$

It has been observed since the findings of Campbell and Woodhead [5] that detonation waves in the real world are almost always unstable. The work of Erpenbeck in 1964 analyzed the stability of steady detonations based on the linearized Euler equations (1)–(3). If only one-dimensional perturbation is considered, for given heat release Q the overdrive f will dictate the stability of the detonations. There is a critical overdrive value f^* such that the steady solution will be unstable if $f < f^*$ and stable otherwise. For $Q = 50$, Erpenbeck predicted $f^* = 1.76$ [8]. Later, in the work of Bourlioux, Majda, and Roytburd [4], f^* was corrected to be 1.73. Erpenbeck provided a neutral stability curve (see the dashed curve in Figure 10 of [8]) based on his eigenmode analysis. It is our objective to calculate the neutral curve by a direct numerical simulation of equations (1)–(3).

3. Numerical methods.

3.1. Multidomain spectral solutions. In this section, we will present a numerical method which could be generalized to a multidimensional simulation of detonation waves. The major difficulties in simulating reacting flow numerically result from disparities between hydrodynamic scales and chemical reaction scales and in the strong dependence of reaction rate on the flow temperature via the Arrhenius rate constant. Traditionally shock capturing schemes were designed to smooth shock. By introducing a considerable amount of numerical viscosity near the shock front, they tend to distort the chemical reaction in the form of unphysical detonations [6] and unintended reactions [7]. With this in mind, we propose to use a multidomain technique as the main framework of our numerical method and tracking technique to accurately represent the shock dynamics without unnecessary numerical viscosities. A similar approach has been used in [13] in the study of the stability of a shock wave interacting with a vortex. In the setting of multidomain approaches, we could apply different types of spatial derivative discretization techniques in different subdomains based on the local properties of the solutions and different time integration techniques. Results in this direction for two-dimensional detonations are reported in [3].

Let $I = [x_{bd}, x_s]$ be the physical domain where the solution is sought. x_{bd} is the left boundary and $x_s = x_s(t)$ is the shock front. The solution domain I will be divided into subdomains I_k,

$$I_k = [a_{k+1}(t), a_k(t)], \qquad k = 1, \dots, M, \tag{5}$$

$$x_{bd} = a_{M+1}(t) < a_M(t) < \cdots < a_2(t) < a_1(t) = x_s(t). \tag{6}$$

The speed of the shock front is $V(t) = \dot{x_s}(t) = \dot{a_1}(t)$. Interfaces $a_1(t), \dots, a_{M+1}(t)$ are generally functions of time t and their evolutions will be determined by corresponding ODEs.

The multidomain spectral solution is denoted as

$$\vec{\mathbf{u}}(x,t) = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} = \begin{pmatrix} \rho \\ \rho u \\ \rho e \\ \rho\lambda \end{pmatrix} \tag{7}$$

and on each domain I_k we define

$$\vec{\mathbf{u}}_{n_k} = \vec{\mathbf{u}}|_{I_k} = \begin{pmatrix} u_1^{(k)} \\ u_2^{(k)} \\ u_3^{(k)} \\ u_4^{(k)} \end{pmatrix}, \tag{8}$$

where n_k is the number of mesh points used in I_k. To compute the spectral solution $\vec{\mathbf{u}}_{n_k}$ on subdomain I_k, we first map I_k into a reference domain $[-1, 1]$,

$$x \in [a_{k+1}(t), a_k(t)] \longrightarrow \xi \in [-1, 1] \tag{9}$$

with

$$\begin{cases} T &= t, \\ \xi &= \xi(x,t) = -1 + 2\frac{x - a_{k+1}(t)}{a_k(t) - a_{k+1}(t)}, \end{cases} \tag{10}$$

or

$$\begin{cases} t &= T, \\ x &= x(\xi, T) = a_{k+1}(t) + \frac{\xi+1}{2}(a_k(t) - a_{k+1}(t)). \end{cases} \tag{11}$$

Let $\{\xi_0^{(k)}, \xi_1^{(k)}, \dots, \xi_{n_k}^{(k)}\}$ be the Chebyshev–Lobatto points on $[-1, 1]$; i.e.,

$$\xi_i^{(k)} = \cos\frac{i\pi}{n_k}, \qquad i = 0, \dots, n_k. \tag{12}$$

Then $x_i^{(k)} = x(\xi_i^{(k)}, T)$ are the mesh points in I_k.

Using the method of lines, the semidiscretized Chebyshev collocation solution $\vec{\mathbf{u}}_{n_k}(x,t)$ for (4) on subdomain I_k satisfies the following set of ODEs for the unknowns $\vec{\mathbf{u}}_{n_k}(x_i^{(k)}, t)$:

$$\tag{13} \left\{ \begin{array}{rcl} \frac{\partial \vec{\mathbf{u}}_{n_k}}{\partial t} + \mathbf{A}(\vec{\mathbf{u}}_{n_k})(\mathcal{I}_{n_k}\vec{\mathbf{u}}_{n_k})_x &=& \Psi(\vec{\mathbf{u}}_{n_k})|_{x=x_i^{(k)}}, \quad 0 \le i \le n_k, \\ \dot{a}_{k+1}(t) &=& V_{k+1}, \\ \dot{a}_k(t) &=& V_k, \\ \dot{V}_{k+1} &=& h_{k+1}, \\ \dot{V}_k &=& h_k, \end{array} \right.$$

where V_k is the speed of interface a_k and $h_k = 0$ if $k \neq 1$. The acceleration of the shock front $h_1 = \dot{V}_1 = \dot{V}(t)$ will be derived in (47). The states of gas in front of x_s are assumed to be given. The states behind $x_{bd} = a_{M+1}$ are assumed to be the steady solution at the Chapman–Jouguet points [11].

The interpolation operator $\mathcal{I}_{n_k}$ is defined on the Chebyshev–Lobatto points $\{\xi_i^{(k)}\}$ on $[-1, 1]$,

$$\tag{14} \mathcal{I}_{n_k}\vec{\mathbf{u}}_{n_k} = \sum_{j=0}^{n_k} \vec{\mathbf{u}}_{n_k}(\xi_j^{(k)})\psi_j(\xi), \qquad \xi \in [-1, 1],$$

where

$$\psi_j(\xi) = \frac{(-1)^{j+1}(1-\xi^2)T'_{n_k}(\xi)}{\bar{c}_j n_k^2(\xi - \xi_j^{(k)})}, \quad \bar{c}_0 = \bar{c}_{n_k} = 2, \quad \bar{c}_j = 1 \quad \text{if } 1 \le j \le n_k - 1.$$

$T_{n_k}(\xi) = \cos(n_k \cos^{-1}(\xi))$ is the Chebyshev polynomial of first kind. Thus

$$\tag{15} \frac{d}{d\xi}\mathcal{I}_{n_k}\vec{\mathbf{u}}_{n_k}(\xi_i^{(k)}) = \sum_{j=0}^{n_k} \vec{\mathbf{u}}_{n_k}(\xi_j^{(k)})\psi'_j(\xi_i^{(k)}).$$

And, finally,

$$\tag{16} (\mathcal{I}_{n_k}\vec{\mathbf{u}}_{n_k})_x|_{x=x_i^{(k)}} = (\mathcal{I}_{n_k}\vec{\mathbf{u}}_{n_k})_\xi(\xi_i^{(k)}, T)\xi_x(x_i^{(k)}, t).$$

We convert all $\frac{\partial}{\partial x}$ and $\frac{\partial}{\partial t}$ derivatives into $\frac{\partial}{\partial \xi}$ and $\frac{\partial}{\partial T}$ derivatives; i.e.,

$$\tag{17} \left\{ \begin{array}{rcl} \frac{\partial}{\partial t} &=& \xi_t(x,t)\frac{\partial}{\partial \xi} + \frac{\partial}{\partial T}, \\ \frac{\partial}{\partial x} &=& \xi_x(x,t)\frac{\partial}{\partial \xi}. \end{array} \right.$$

It follows from (13) that $\vec{\mathbf{u}}_{n_k}(\xi, T)$ satisfies

$$\tag{18} \left\{ \begin{array}{rcl} \frac{\partial \vec{\mathbf{u}}_{n_k}}{\partial T} + [\mathbf{A}(\vec{\mathbf{u}}_{n_k})\xi_x + \xi_t](\mathcal{I}_{n_k}\vec{\mathbf{u}}_{n_k})_\xi &=& \Psi(\vec{\mathbf{u}}_{n_k})|_{\xi=\xi_i^{(k)}}, \\ \dot{a}_k &=& V_k, \\ \dot{a}_{k+1} &=& V_{k+1}, \\ \dot{V}_k &=& h_k, \\ \dot{V}_{k+1} &=& h_{k+1}. \end{array} \right.$$

The stability of multidomain spectral solutions by (18) for $k = 1, \ldots, M$ depends on the treatment of values on the interface between neighboring subdomains. We discuss the procedure for calculating those values in the following subsection.

3.2. Characteristic correction for interface treatment. Consider subdomains $I_{k+1} = [a_{k+2}, a_{k+1}]$ and $I_k = [a_{k+1}, a_k]$ with a common interface at a_{k+1}. For the simplicity of illustration, we assume that a fully discretized version of (18) has been obtained by using the Euler forward method for the time derivative; therefore, we have

$$\vec{\mathbf{u}}^{n+1}_{n_k,i} = \vec{\mathbf{u}}^{n}_{n_k,i} - \Delta T[\mathbf{A}(\vec{\mathbf{u}}_{n_k,i})\xi_x + \xi_t](\mathcal{I}_{n_k}\vec{\mathbf{u}}^{n}_{n_k})_\xi|_{\xi=\xi_i^{(k)}} + \Delta T\Phi(\vec{\mathbf{u}}^{n}_{n_k,i}), \tag{19}$$

where $\vec{\mathbf{u}}_{n_k,i} = \vec{\mathbf{u}}_{n_k}(\xi_i^{(k)}, T^n)$ and T^n is the discretized time with $\Delta T = T^{n+1} - T^n$.

Similarly, on subdomain I_{k+1} we have

$$\vec{\mathbf{u}}^{n+1}_{n_{k+1},i} = \vec{\mathbf{u}}^{n}_{n_{k+1},i} - \Delta T[\mathbf{A}(\vec{\mathbf{u}}_{n_{k+1},i})\xi_x + \xi_t](\mathcal{I}_{n_{k+1}}\vec{\mathbf{u}}^{n}_{n_{k+1}})_\xi|_{\xi=\xi_i^{(k+1)}} + \Delta T\Phi(\vec{\mathbf{u}}^{n}_{n_{k+1},i}), \tag{20}$$

where $\vec{\mathbf{u}}_{n_{k+1},i} = \vec{\mathbf{u}}_{n_{k+1}}(\xi_i^{(k+1)}, T^n)$.

Therefore, from (19) and (20), two solution vectors will be computed on interface a_{k+1}; i.e.,

$$\vec{\mathbf{u}}^L = \vec{\mathbf{u}}_{n_{k+1}}(x_0^{(k+1)}), \qquad \vec{\mathbf{u}}^R = \vec{\mathbf{u}}_{n_k}(x_{n_k}^{(k)}). \tag{21}$$

For $\vec{\mathbf{u}}$ to be continuous at interface a_{k+1}, $\vec{\mathbf{u}}_{n_k}, \vec{\mathbf{u}}_{n_{k+1}}$ should have the same value. Also, any choice of this value should be based on the characteristic information of the hyperbolic systems (4) [12].

Let $\vec{\mathbf{u}}_{\text{roe}}(a_{k+1})$ be the Roe-average state between $\vec{\mathbf{u}}^L$ and $\vec{\mathbf{u}}^R$ [15]. Using this $\vec{\mathbf{u}}_{\text{roe}}(a_{k+1})$, we compute the Jacobian of flux $\mathbf{F}(\vec{\mathbf{u}})$ in (2) as

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ \alpha_1 & \alpha_2 & (\gamma - 1) & \alpha_3 \\ \beta_1 & \beta_2 & \gamma u & \beta_3 \\ -\lambda u & \lambda & 0 & u \end{pmatrix}, \tag{22}$$

where

$$\begin{aligned} \alpha_1 &= \tfrac{\gamma-3}{2}u^2, \\ \alpha_2 &= (3-\gamma)u, \\ \alpha_3 &= -(\gamma - 1)Q, \\ \beta_1 &= -\gamma u e + (\gamma - 1)(u^2 + \lambda u Q), \\ \beta_2 &= \gamma e - \tfrac{3(\gamma-1)}{2}u^2 - (\gamma - 1)\lambda Q, \\ \beta_3 &= -(\gamma - 1)u, \end{aligned}$$

and e is the internal energy

$$e = \frac{1}{\gamma - 1}\frac{p}{\rho} + \frac{u^2}{2} + \lambda Q.$$

Matrix $\mathbf{A}$ has four eigenvalues

$$\mu_1 = u - c, \quad \mu_2 = \mu_3 = u, \quad \mu_4 = u + c, \tag{23}$$

where $c = \sqrt{\gamma(p/\rho)}$ is the sound speed. The four left and right eigenvectors are $\mathbf{l}^{(k)}$ and $\mathbf{r}^{(k)}$, $1 \le k \le 4$,

$$\mathbf{l}^{(1)} = \frac{1}{2}\left(b_2 + \frac{u}{c}, -ub_1 - \frac{1}{c}, b_1, -Qb_1\right), \tag{24}$$

$$\mathbf{l}^{(2)} = (-\lambda b_2, \lambda u b_1, -\lambda b_1, 1 + \lambda b_1 Q), \tag{25}$$

$$\mathbf{l}^{(3)} = (1 - \theta b_2, \theta u b_1, -\theta b_1, -1 + \theta b_1 Q), \tag{26}$$

$$\mathbf{l}^{(4)} = \frac{1}{2}\left(b_2 - \frac{u}{c}, -u b_1 + \frac{1}{c}, b_1, -Q b_1\right), \tag{27}$$

and

$$\mathbf{r}^{(1)} = (1, u - c, h - uc, \lambda)^{\mathrm{T}}, \tag{28}$$

$$\mathbf{r}^{(2)} = \left(1, u, \frac{u^2}{2} + Q, 1\right)^{\mathrm{T}}, \tag{29}$$

$$\mathbf{r}^{(3)} = \left(1, u, \frac{u^2}{2}, 0\right)^{\mathrm{T}}, \tag{30}$$

$$\mathbf{r}^{(4)} = (1, u + c, h + uc, \lambda)^{\mathrm{T}}, \tag{31}$$

where $\theta = 1 - \lambda$, $b_1 = \frac{\gamma - 1}{c^2}$, $b_2 = \frac{u^2}{2} b_1$, and $h = c^2 + \frac{u^2}{2} - \lambda Q$.

It is easy to check that

$$\mathbf{l}^{(p)} \circ \mathbf{r}^{(q)} = \delta_{p,q}, \qquad p, q = 1, \ldots, 4. \tag{32}$$

Now define the left eigenvector matrix $\mathbf{P}$:

$$\mathbf{P} = \begin{pmatrix} \mathbf{l}^{(1)} \\ \mathbf{l}^{(2)} \\ \mathbf{l}^{(3)} \\ \mathbf{l}^{(4)} \end{pmatrix}; \tag{33}$$

then

$$\mathbf{P}^{-1} = (\mathbf{r}^{(1)}, \mathbf{r}^{(2)}, \mathbf{r}^{(3)}, \mathbf{r}^{(4)}).$$

We define both characteristic decompositions for $\vec{\mathbf{u}}^L$, $\vec{\mathbf{u}}^R$ as

$$\vec{w}^L = \mathbf{P}\vec{\mathbf{u}}^L = \begin{pmatrix} w_1^L \\ w_2^L \\ w_3^L \\ w_4^L \end{pmatrix}, \qquad \vec{w}^R = \mathbf{P}\vec{\mathbf{u}}^R = \begin{pmatrix} w_1^R \\ w_2^R \\ w_3^R \\ w_4^R \end{pmatrix}. \tag{34}$$

And common characteristic values on the interface should be assigned on interface a_{k+1} based on the signs of $\mu_i - \dot{a}_{k+1}$; i.e., for $1 \le i \le 4$,

$$w_i = \begin{cases} w_i^R & \text{if } \mu_i - \dot{a}_{k+1} \le 0, \\ w_i^L & \text{otherwise.} \end{cases} \tag{35}$$

Finally, we set $\vec{w} = (w_1, w_2, w_3, w_4)^{\mathrm{T}}$ and

$$\vec{\mathbf{u}}^L = \vec{\mathbf{u}}^R = \vec{\mathbf{u}}(a_{k+1}) \equiv \mathbf{P}^{-1}\vec{w}. \tag{36}$$

3.3. The tracking algorithm for the shock front. To complete system (18), we need to give an ODE for shock speed $V(t) = V_1(t)$. Here a higher-order tracking algorithm will be used for the front shock; a previous attempt to track detonation waves can be found in [2]. We start by using the Rankine–Hugoniot condition across the shock front; i.e.,

$$\left\{\begin{array}{rcl} \rho_0(u_0 - V) &=& \rho_1(u_1 - V), \\ \rho_0(u_0 - V)^2 + p_0 &=& \rho_1(u_1 - V)^2 + p_1, \\ \frac{\gamma}{\gamma-1}\frac{p_0}{\rho_0} + \frac{1}{2}(u_0 - V)^2 &=& \frac{\gamma}{\gamma-1}\frac{p_1}{\rho_1} + \frac{1}{2}(u_1 - V)^2, \\ \lambda_0 &=& \lambda_1, \end{array}\right. \tag{37}$$

where “0” denotes the states in front of the shock and “1” the states behind the shock.

Solving the quantities behind the shock in terms of those in front of the shock, we get

$$\left\{\begin{array}{rcl} \rho_1 &=& \rho_0\frac{\frac{\gamma+1}{\gamma-1}M_0^2}{\frac{2}{\gamma-1}+M_0^2}, \\ u_1 &=& u_0 - (u_0 - V)(1 - \frac{\rho_0}{\rho_1}), \\ p_1 &=& p_0 + \rho_0(u_0 - V)^2(1 - \frac{\rho_0}{\rho_1}), \\ \lambda_1 &=& \lambda_0, \end{array}\right. \tag{38}$$

where mach number $M_0 = \frac{|u_0-V|}{c_0}$, $c_0 = \sqrt{\gamma(p_0/\rho_0)}$.

Define the derivative following the motion of the shock front $\frac{D}{Dt} = \frac{\partial}{\partial t} + V(t)\frac{\partial}{\partial x}$ and differentiate both sides of (38) to get

$$\begin{array}{rcl} \frac{D\rho_1}{Dt} &=& E_1\dot{V} + F_1, \\ \frac{Du_1}{Dt} &=& E_2\dot{V} + F_2, \\ \frac{Dp_1}{Dt} &=& E_3\dot{V} + F_3, \\ \frac{D\lambda_1}{Dt} &=& E_4\dot{V} + F_4, \end{array} \tag{39}$$

where

$$\begin{array}{ll} E_1 = -\frac{4\rho}{\gamma-1}\frac{1}{(u_0-V)(\frac{2}{\gamma-1}+M_0^2)}, & E_2 = \frac{2(1+M_0^2)}{(\gamma+1)M_0^2}, \\ E_3 = -4\rho_0\frac{(u_0-V)}{\gamma+1}, & E_4 = 0, \end{array} \tag{40}$$

and

$$\begin{array}{l} F_1 = \frac{\rho}{\rho_0}\frac{D\rho_0}{Dt} + \frac{2\rho_0\frac{\gamma+1}{(\gamma-1)^2}}{(\frac{2}{\gamma-1}+M_0^2)^2}(\frac{2(u_0-V)}{c_0^2}\frac{Du_0}{Dt} - \frac{M_0^2}{c_0^2}\frac{Dc_0^2}{Dt}), \\ F_2 = \frac{M_0^2-\frac{2}{\gamma-1}}{\frac{\gamma+1}{\gamma-1}M_0^2}\frac{Du_0}{Dt} + \frac{2}{(\gamma+1)(u_0-V)}\frac{Dc_0^2}{Dt}, \\ F_3 = (u_0 - V)^2(1 - \frac{\rho_0}{\rho})\frac{D\rho_0}{Dt} + \frac{4\rho_0(u_0-V)}{\gamma+1}\frac{Du_0}{Dt} - \frac{2\rho_0}{\gamma+1}\frac{Dc_0^2}{Dt} + \frac{Dp_0}{Dt}, \\ F_4 = \frac{D\lambda_0}{Dt}. \end{array} \tag{41}$$

Rewriting (39) for the conserved variables $\rho_1, \rho_1 u_1, \rho e_1, \rho_1\lambda_1$, we get

$$\begin{array}{rcl} \frac{D\rho_1}{Dt} &=& G_1\dot{V} + H_1, \\ \frac{D\rho_1 u_1}{Dt} &=& G_2\dot{V} + H_2, \\ \frac{D\rho_1 e_1}{Dt} &=& G_3\dot{V} + H_3, \\ \frac{D\rho_1\lambda_1}{Dt} &=& G_4\dot{V} + H_4, \end{array} \tag{42}$$

where

$$G_1 = E_1, \qquad G_2 = u_1 E_1 + \rho_1 E_2,$$
$$G_3 = (\tfrac{1}{2}u_1^2 + \lambda_1 Q)E_1 + \rho_1 u_1 E_2 + \tfrac{1}{\gamma-1}E_3, \quad G_4 = \lambda_1 E_1,$$

and

$$H_1 = F_1, \qquad H_2 = u_1 F_1 + \rho_1 F_2,$$
$$H_3 = (\tfrac{1}{2}u_1^2 + \lambda_1 Q)F_1 + \rho_1 u_1 F_2 + \rho_1 Q F_4, \quad H_4 = \lambda_1 F_1 + \rho_1 F_4.$$

To derive the ODE for the shock speed $V(t)$, we consider the characteristic form of (18) which is obtained by multiplying (18) with matrix **P** (subscript n_k omitted):

$$\mathbf{P}\frac{\partial \vec{\mathbf{u}}}{\partial T} + \mathbf{P}[\mathbf{A}\xi_x + \xi_t]\frac{\partial \mathcal{I}\vec{\mathbf{u}}}{\partial \xi} = \mathbf{P}\Psi(\vec{\mathbf{u}}). \tag{43}$$

Now using the fact that $\mathbf{P}A\mathbf{P}^{-1} = \operatorname{diag}(u-c, u, u, u+c)$, we have

$$\mathbf{P}\frac{\partial \vec{\mathbf{u}}}{\partial T} + \left[\begin{pmatrix} u-c & & & \\ & u & & \\ & & u & \\ & & & u+c \end{pmatrix}\xi_x + \xi_t\right]\mathbf{P}\frac{\partial \mathcal{I}\vec{\mathbf{u}}}{\partial \xi} = \mathbf{P}\Psi(\vec{\mathbf{u}}). \tag{44}$$

For a strong detonation, the flow is subsonic behind the shock front $x_s(t)$. Therefore, $u + c > V(t) = \dot{a}_1(t)$ at the shock front. Thus the fourth equation of (44) corresponds to an incoming characteristic toward the shock front. Meanwhile, it is easy to verify that on the shock front $x_s(t)$, $\frac{\partial}{\partial T} = \frac{D}{Dt}$. Now denoting the last row of **P** as $(P_{4,1}, P_{4,2}, P_{4,3}, P_{4,4})$, the fourth equation of (44) reads

$$P_{4,1}\frac{D\rho_1}{Dt} + P_{4,2}\frac{D\rho_1 u_1}{Dt} + P_{4,3}\frac{D\rho_1 e_1}{Dt} + P_{4,4}\frac{D\rho_1\lambda_1}{Dt} = W, \tag{45}$$

$$W = P_{4,1}R_1 + P_{4,2}R_2 + P_{4,3}R_3 + P_{4,4}R_4, \tag{46}$$

where $\mathbf{R} = (R_1, R_2, R_3, R_4) = \Psi(\vec{\mathbf{u}}) - (\mathbf{A}\xi_x + \xi_t)\frac{\partial \mathcal{I}\vec{\mathbf{u}}}{\partial \xi}$ is the residual of solution $\vec{\mathbf{u}}$ which will be computed by the Chebyshev spectral collocation method.

Finally (42) and (45) give the ODE for the shock speed

$$\dot{V}(t) = \frac{W - (P_{4,1}H_1 + P_{4,2}H_2 + P_{4,3}H_3 + P_{4,4}H_4)}{P_{4,1}G_1 + P_{4,2}G_2 + P_{4,3}G_3 + P_{4,4}G_4}. \tag{47}$$

4. Calculation of neutral stability curve. From linear analysis [4] and [8], instability of detonation waves is associated with the eigenmodes of the linearized system whose eigenvalues have positive real parts. Disturbances containing those modes will grow exponentially in time. However, when we are close to the neutral stability limit, the growth of such disturbances will be slow. It demands highly accurate numerical techniques to distinguish the unstable and stable detonations. When we identify the stability limit based on direct numerical calculations, the accuracy of critical overdrive f^* cannot be expected to be more accurate than the truncation errors of the methods. This is why it is crucial to use highly accurate methods in studying neutral stability limits.

Convergence studies of numerical methods. Pressure history at the von Neumann spikes will be recorded at equally spaced time stations and used to determine stable or unstable detonations based on its decay or growth. When the overdrive is close to the neutral stability

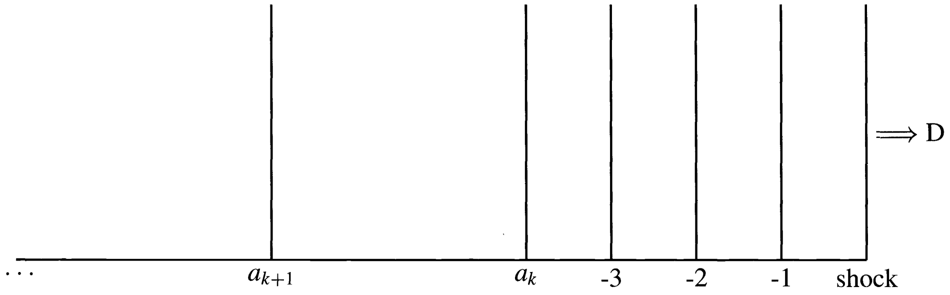


FIG. 1. *Multidomain set up. All interfaces are moving in time.*

limit f^*, it is difficult to tell visually the growth or decay of different frequencies from the pressure history. Therefore, we employ discrete Fourier transformation to analyze the growth or decay of all the frequencies. As for an unstable detonation, growth in magnitudes of initial disturbances (truncation error in our methods) will depend on the mesh size of the particular simulation. Therefore, we will not use the magnitudes of various frequencies in the pressure history as a checkup for the accuracy of mesh resolutions. Instead, we will be concerned with the unstable frequencies produced by different meshes and determine the accuracy of the numerical solution based on the convergence of the unstable frequencies.

Discrete Fourier transformation. Suppose we have N consecutive sampled values $h_k = h(t_k)$, $t_k = k\Delta_t$, $k = 0, 1, 2, \ldots, N-1$. The discrete Fourier transform is defined by

$$H(f_n) = \Delta_t \sum_{k=0}^{N-1} h_k \exp\left(\frac{2\pi ikn}{N}\right),$$

where $f_n = \frac{n}{N\Delta_t}$, $n = -\frac{N}{2}, -\frac{N}{2}+1, \ldots, \frac{N}{2}-1, \frac{N}{2}$. $H(f_n)$ gives the phase and magnitude of the wave whose frequency is f_n. When the time series data h_k are all real-valued, a symmetric property holds: $H(-f) = H(f)^*$, where * denotes the complex conjugate. Thus $2|H(f_n)|$ is the magnitude of the wave whose frequency is f_n, for $n = 1, 2, \ldots, \frac{N}{2}$.

We test the accuracy of the numerical method in §3 for the case $Q = 50$, $f = 1.6$, which has been identified as an unstable detonation [4]. The mesh setup is depicted in Figure 1.

The number of initial subdomains is nine and the interfaces will move with the average speed of the shock front except for the shock front itself whose speed is given by (47). More subdomains will be added to the rear part of the computational domain. We monitor the solutions in the last subdomains so that no waves will be interacting with the leftmost subdomain boundaries. The boundary conditions to the left of the computational domains are the steady solutions at the Chapman–Jouguet point. The initial conditions will be the Chapman–Jouguet steady detonation. Therefore, we are simulating a one-dimensional detonation wave travelling in an infinitely long tube and the driven piston is far away from the region where computations are done.

In Figure 2, results for two meshes are given; the coarse mesh has five points per half reaction length interval while the fine mesh has ten points per half reaction length interval. On the top, pressure histories are given for both runs at a time interval $\Delta_t = 0.1$. At the bottom, the Fourier transformations of the output pressure data from time $t = 140$ to $t = 200$ are given. We intentionally ignore the early time history to wait for the full development of the pressure oscillations. We can see clearly in each case that we have various frequencies. In

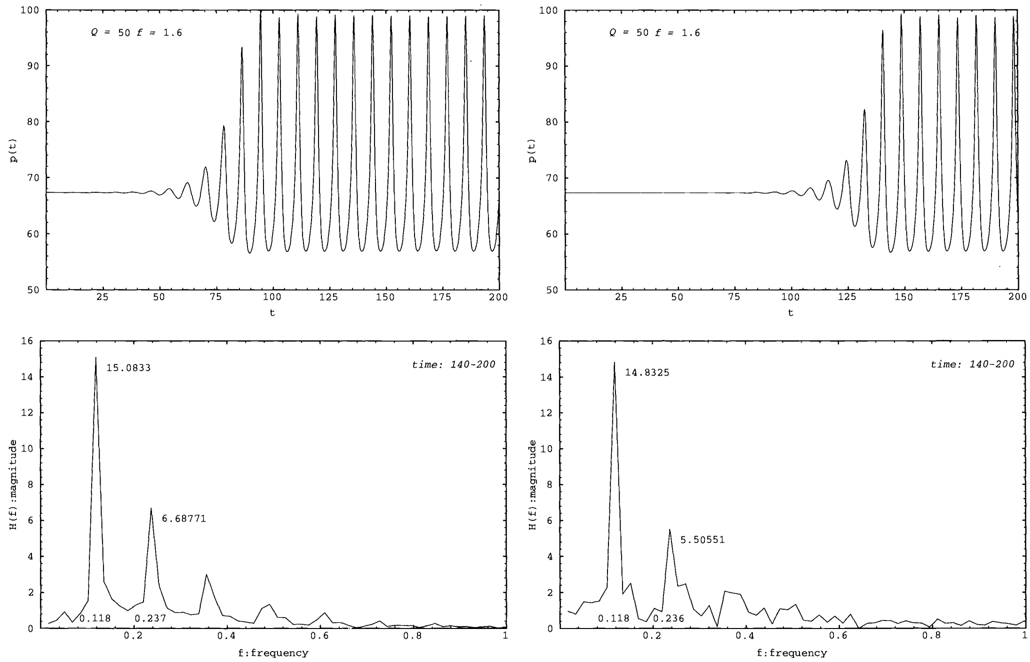


FIG. 2. $Q = 50$, $f = 1.6$. *(Left) coarse mesh—five points per half reaction length, (Right) fine mesh—10 points per half reaction length. Top plots are the pressure histories and bottom plots are their Fourier transformations for time* $t = 140$–200.

TABLE 1

Convergence tests: frequencies and magnitudes of Fourier transformations for the pressure history. Coarse mesh $5/L_{1/2}$, *fine mesh* $10/L_{1/2}$.

N	f_1		f_2	
	Location	Magnitude	Location	Magnitude
$5/L_{1/2}$	0.1184493	15.083265	0.2368986	6.6877127
$10/L_{1/2}$	0.1181893	14.832513	0.2363786	5.5055077

Table 1, we report the locations of the first two frequencies with the largest magnitudes, i.e., f_1, f_2. The accuracy for the locations of both frequencies are up to the third decimal digit.

This shows that using our methods, five points per half reaction length interval will be enough to achieve satisfactory accuracy. Still, in all our computations of critical overdrives f^*, we have used ten points per half reaction length interval for subdomains near the shock front. We use the explicit second-order Runge–Kutta method for the time discretizations. Finally, Figure 3 depicts the solution of pressure and density and mass fraction using the fine mesh (ten points per half reaction length) at time $t = 145$.

Neutral stability curve. In Table 2, we give the approximation of critical overdrive f^* for 16 heat releases Q. For each Q, we compute the largest overdrive (denoted as f^-) found to produce unstable detonation and the smallest overdrive (denoted as f^+) found to produce stable detonation. Then we conclude that an approximation for f^* can be obtained by averaging f^+ and f^-; i.e., $f^* = \frac{f^+ + f^-}{2}$.

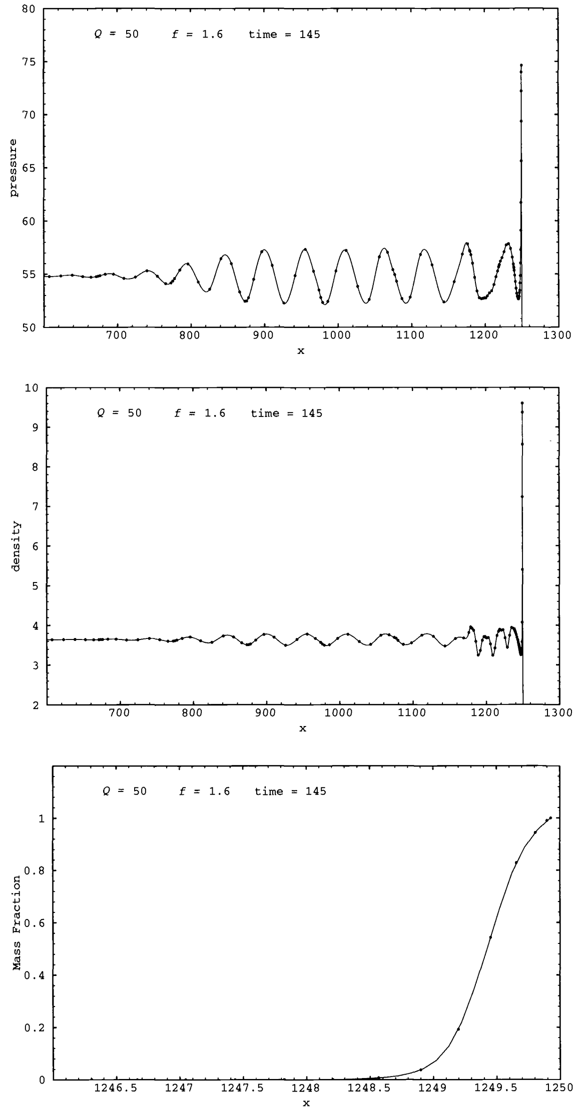


FIG. 3. $Q = 50$, $f = 1.6$. *Solutions of pressure and density and mass fraction of reactant (from top to bottom) on fine mesh (*10 *points per half reaction) at time* $t = 145$.

For a given heat release Q, in order to decide which overdrive gives stable or unstable detonation, we record the pressure $p(t)$ at the von Neumann spikes up to time $t = 150$. Then we compute two consecutive Fourier transformations of $p(t) - p_0$ (p_0 as the von Neumann pressure of the steady detonation), one for the time period [50, 100] and one for the time period [100,150]. Comparing the magnitudes of corresponding frequencies, if all frequencies are found to be decreasing, then the detonation will be considered stable. If the magnitude of any of the frequencies is found to have increased, then the detonation will be considered unstable.

TABLE 2
Critical values of overdrive for selected Q's.

Q	f^- Largest unstable overdrive	f^+ Smallest stable overdrive	$f^* = (f^- + f^+)/2$ Critical overdrive
0.5	1.120	1.125	1.1225
0.6	1.255	1.260	1.2575
0.7	1.395	1.400	1.3975
0.9	1.685	1.690	1.6875
1.0	1.825	1.830	1.8275
1.2	2.205	2.210	2.2075
1.4	2.505	2.510	2.5075
1.7	2.845	2.850	2.8475
2.0	3.080	3.085	3.0825
2.5	3.335	3.340	3.3375
5.0	3.615	3.620	3.6175
10.0	3.250	3.250	3.2525
15.0	2.865	2.870	2.8675
20.0	2.585	2.590	2.5875
30.0	2.200	2.205	2.2025
50.0	1.730	1.731	1.7305

In Figure 4, $Q = 1$; the left column is for the unstable case $f^- = 1.825$. The top left is the time history of the pressure perturbation $p'(t)$ in reference to the steady state pressure; i.e., $p'(t) = p(t) - p_0$. As the overdrive f^+ and f^- are close to the critical value, the pressure perturbation $p'(t)$ is very small. For display purposes, the scale for the history of pressure perturbation $p'(t)$ has been scaled up by a factor of 1000. (This scaling will be used for all pressure perturbation histories thereafter.) The left middle plot is the Fourier transformation of $p'(t)$ between time $t = 50$ and time $t = 100$. We can see there are two basic frequencies, which we denote as $f_1 = 0.12$ and $f_2 = 0.44$ in an increasing order. The magnitudes for f_1, f_2 are $H(f_1) = 0.00294285$, $H(f_2) = 0.00399988$, respectively. The left bottom plot is the Fourier transformation of $p'(t)$ from time $t = 100$ to time $t = 150$. Again we see two basic frequencies $f_1 = 0.12$, $f_2 = 0.44$. However, the magnitudes for f_1, f_2 are $H(f_1) = 0.00319032$, $H(f_2) = 0.0024562$, respectively. Comparing the magnitude of $H(f_1), H(f_2)$ in these Fourier transformations for two consecutive time windows, we find that the magnitude of the first frequency $H(f_1)$ has increased, while the magnitude for the second frequency $H(f_2)$ has decreased. Therefore, we conclude that $f^+ = 1.825$ gives unstable detonation. Meanwhile, in the right column of Figure 4, there are three plots for overdrive $f^+ = 1.830$. The top right is the time history of pressure perturbation $p'(t)$ and the middle and bottom right are the Fourier transformations of $p'(t)$ for the time periods [50, 100] and [100, 150], respectively. Again we found two basic frequencies f_1, f_2. For the time period [50, 100], $H(f_1) = 0.00267143$, $H(f_2) = 0.00317801$, while for the time period [100, 150], $H(f_1) = 0.00266544$, $H(f_2) = 0.00169358$. It is found that the magnitudes for both frequencies have decreased over the two consecutive time windows and, therefore, we conclude that $f^+ = 1.830$ gives a stable detonation. By taking the average of f^+ and f^-, we have an approximation for the critical overdrive $f^* = 1.8275$ when $Q = 1$.

In Figure 5, we present the case of $Q = 50$. On the Fourier transformation, only one basic frequency is found in this case. $f^- = 1.730$ is the largest overdrive which gives an unstable detonation, and $f^+ = 1.731$ is the smallest overdrive that gives a stable detonation. Therefore, the critical overdrive f^* can be taken approximately as $f^* = 1.7305$ when $Q = 50$.

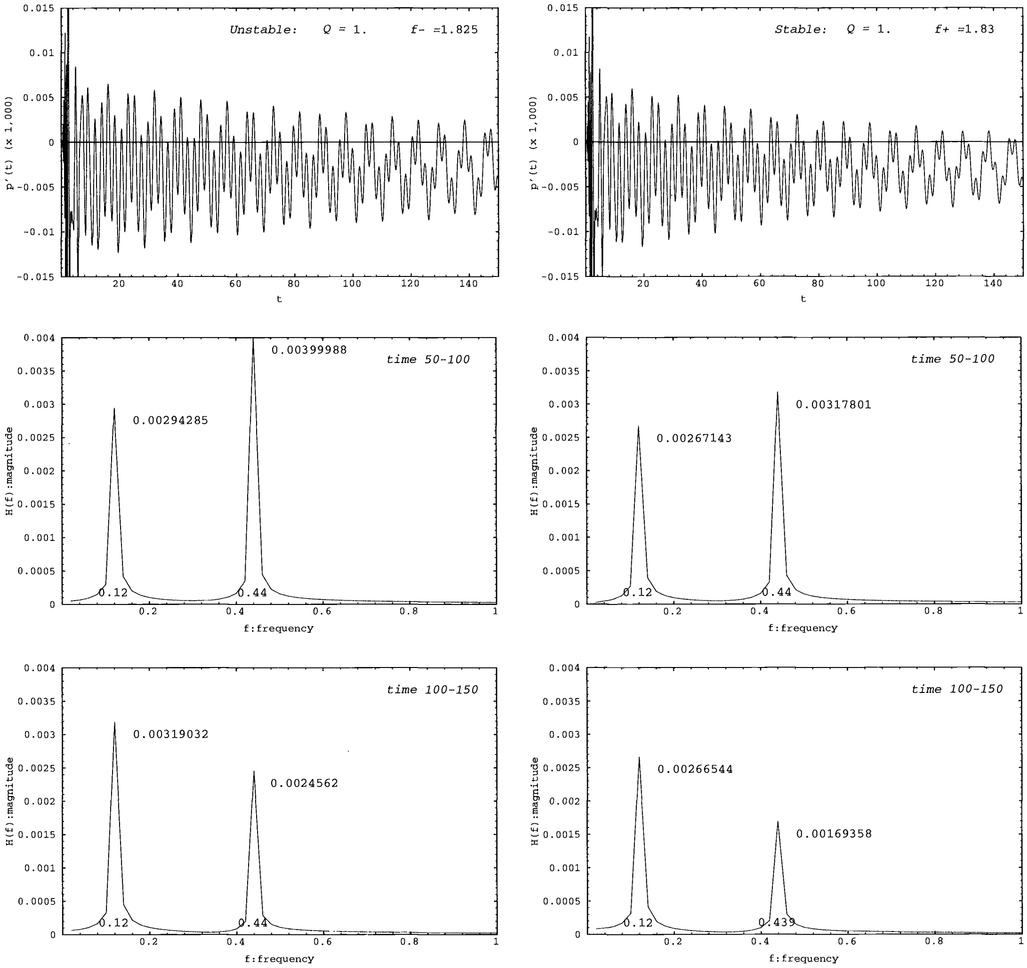


FIG. 4. $Q = 1.0$. *(Left)* $f^- = 1.825$, *unstable, pressure perturbation history (top), Fourier frequencies for* $50 \le t \le 100$ *(middle), and* $100 \le t \le 150$ *(bottom). (Right)* $f^+ = 1.830$, *stable, pressure perturbation history (top), Fourier frequencies for* $50 \le t \le 100$ *(middle), and* $100 \le t \le 150$ *(bottom).*

In Table 3, we have listed Q's for all sixteen heat releases for the first two frequencies and the magnitudes of Fourier transformations of $p'(t)$ over two time periods, i.e., one for the time [50, 100] and one for the time [100, 150]. (Again, all $p'(t)$ have been scaled up by a factor of 1000.)

Finally, in Figure 6, we plot the neutral stability curve based on the data in Table 2; the solid curve is obtained by a cubic Hermite spline interpolation. Note that the heat release Q-axis is logarithmic. The smaller dots and the dashed curve are the results of Erpenbeck using eigenmode analysis [8].

5. Conclusion. We have calculated the neutral stability curve for one-dimensional detonation waves using a highly accurate multidomain shock tracking algorithm. The study of neutral stability will provide insight into more complex studies of multidimensional stability problems. However, it should be realized that the accuracy of results based on direct numerical simulation is limited by the inherent truncation errors of the numerical methods.

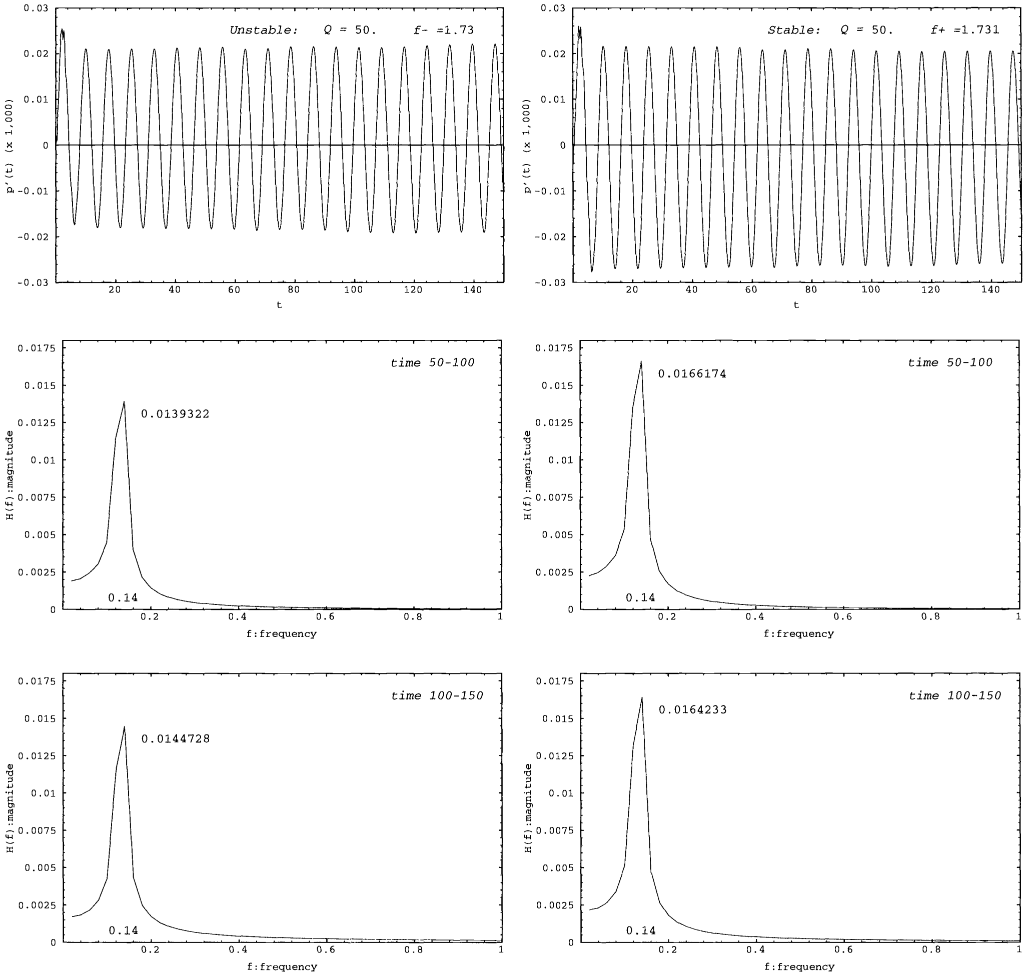


FIG. 5. $Q = 50.0$. *(Left)* $f^{-} = 1.730$, *unstable, pressure perturbation history (top), Fourier frequencies for* $50 \leq t \leq 100$ *(middle), and* $100 \leq t \leq 150$ *(bottom). (Right)* $f^{+} = 1.731$, *stable, pressure perturbation history (top), Fourier frequencies for* $50 \leq t \leq 100$ *(middle), and* $100 \leq t \leq 150$ *(bottom).*

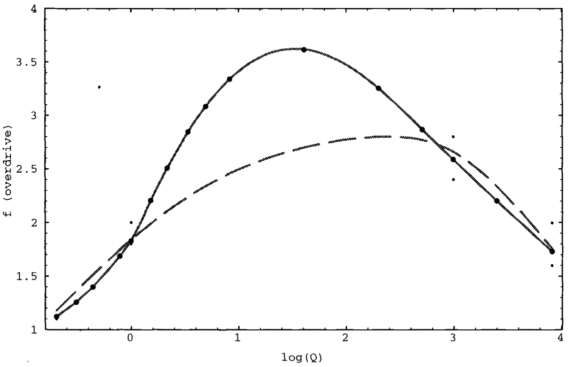


FIG. 6. *Neutral stability curve. Larger dots and solid line are based on data in Table* 3 *and smaller dots and dashed line are from Erpenbeck* [8].

TABLE 3

Frequencies and magnitudes of Fourier transformations of pressure history $p'(t)$ for unstable and stable overdrives f^- and f^+. An asterisk $\ast$ indicates nonexistence of the corresponding frequency.

Q			f^-				f^+			
			$50 \le t \le 100$		$100 \le t \le 150$		$50 \le t \le 100$		$100 \le t \le 150$	
	f_1	f_2	$H(f_1)$	$H(f_2)$	$H(f_1)$	$H(f_2)$	$H(f_1)$	$H(f_2)$	$H(f_1)$	$H(f_2)$
0.5	0.06	*	0.000338	*	0.000378	*	0.000328	*	0.000321	*
0.6	0.08	*	0.000116	*	0.000127	*	0.000095	*	0.000092	*
0.7	0.10	*	0.000190	*	0.000202	*	0.000175	*	0.000165	*
0.9	0.12	*	0.002533	*	0.002602	*	0.002175	*	0.002044	*
1.0	0.12	0.44	0.002942	0.003999	0.003190	0.002456	0.002671	0.003178	0.002665	0.001693
1.2	0.14	0.50	0.000768	0.004174	*	0.004463	0.007146	0.003079	*	0.002862
1.4	0.14	0.52	0.000474	0.001457	*	0.001562	0.000432	0.001172	*	0.001085
1.7	*	0.56	*	0.006647	*	0.006768	*	0.005404	*	0.004731
2.0	*	0.58	*	0.011201	*	0.011666	*	0.008805	*	0.007856
2.5	*	0.60	*	0.015564	*	0.015697	*	0.012337	*	0.010762
5.0	*	0.64	*	0.021898	*	0.022085	*	0.016842	*	0.013993
10.	0.16	0.66	0.003955	0.022479	*	0.028927	0.003552	0.015466	*	0.015421
15.	0.16	0.68	0.015520	0.014264	0.011176	0.016416	0.013404	0.008533	0.008632	0.007315
20.	0.14	*	0.022994	*	0.023831	*	0.019039	*	0.017071	*
30.	0.14	*	0.008826	*	0.009254	*	0.007482	*	0.006721	*
50.	0.14	*	0.013932	*	0.014472	*	0.016617	*	0.016423	*

REFERENCES

[1] G. E. ABOUSEIF AND T. Y. TOONG, *Theory of unstable one-dimensional detonations*, Combustion and Flame, 45 (1982), pp. 67–94.

[2] B. BUKIET, *Application of front tracking to two dimensional curved detonation fronts*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 80–99.

[3] W. CAI, *High order numerical methods for the computation of cellular structures of* 2-D *detonation waves*, AIAA J., 33 (1995), pp. 1248–1255.

[4] A. BOURLIOUX, A. J. MAJDA, AND V. ROYTBURD, *Theoretical and numerical structures for unstable one-dimensional detonations*, SIAM J. Appl. Math., 51 (1991), pp. 303–343.

[5] C. CAMPBELL AND D. W. WOODHEAD, J. Chem. Soc., 125 (1926), p. 3010; 127 (1927), p. 1572.

[6] P. COLELLA, A. MAJDA, AND V. ROYTBURD, *Theoretical and numerical structure for reacting shock waves*, SIAM J. Sci. Statist. Comput., 4 (1986), pp. 1059–1080.

[7] B. E. ENGQUIST AND B. SJOGREEN, *Robust Difference Approximations of Stiff Inviscid Detonation Waves*, CAM REPORT 91-03, University of California, Los Angeles, 1991.

[8] J. ERPENBECK, *Stability of idealized one-reaction detonations*, Phys. Fluids, 7 (1964), pp. 684–696.

[9] W. FICKETT AND W. W. WOOD, *Flow calculation for pulsating one-dimensional detonations*, Phys. Fluids, 9 (1966), pp. 903–916.

[10] W. FICKETT, J. D. JACOBSON, AND G. L. SCHOTT, *Calculated pulsating one-dimensional detonations with induction-zone kinetics*, AIAA J., 10 (1972), pp. 514–516.

[11] W. FICKETT AND W. C. DAVIS, *Detonation*, University of California Press, Berkeley, CA 1979.

[12] D. GOTTLIEB, M. GUNZBURGER, AND E. TURKEL, *On numerical boundary treatment for hyperbolic systems*, SIAM J. Numer. Anal., 19 (1982), pp. 671–697.

[13] D. A. KOPRIVA, *A multi-domain spectral collocation computation of the sound generated by a shock-vortex interaction*, in Computational Acoustic and Wave Propagation, M. Shultz, D. Lee Sternberg, eds., North-Holland, Amsterdam, 1988.

[14] E. S. ORAN, J. P. BORIS, M. FLANIGAN, T. BURKS, AND M. PICONE, *Numerical simulations of detonations in hydrogen-air and methane-air mixtures*, Eighteenth International Symposium on Combustion, The Combustion Institute, Washington, DC, 1981, pp. 1641–1649.

[15] P. L. ROE, *Approximate Riemann solvers, parameters vectors, and difference schemes*, J. Comput. Phys., 43 (1981), pp. 357–372.

[16] S. TAKI AND T. FUJIWARA, *Numerical analysis of two dimensional nonsteady detonations*, AIAA J., 16 (1978), pp. 73–77.

SIAM J. SCI. COMPUT.
Vol. 17, No. 4, pp. 830–847, July 1996

ILUM: A MULTI-ELIMINATION ILU PRECONDITIONER FOR GENERAL SPARSE MATRICES*

Y. SAAD†

Abstract. Standard preconditioning techniques based on incomplete LU (ILU) factorizations offer a limited degree of parallelism, in general. A few of the alternatives advocated so far consist of either using some form of polynomial preconditioning or applying the usual ILU factorization to a matrix obtained from a multicolor ordering. In this paper we present an incomplete factorization technique based on independent set orderings and multicoloring. We note that in order to improve robustness, it is necessary to allow the preconditioner to have an arbitrarily high accuracy, as is done with ILUs based on threshold techniques. The ILUM factorization described in this paper is in this category. It can be viewed as a multifrontal version of a Gaussian elimination procedure with threshold dropping which has a high degree of potential parallelism. The emphasis is on methods that deal specifically with general unstructured sparse matrices such as those arising from finite element methods on unstructured meshes.

Key words. sparse linear systems, preconditioned Krylov subspace methods, incomplete LU factorizations, independent set orderings, multicoloring, graph coloring, threshold dropping strategies

AMS subject classification. 65F10

1. Introduction. In this paper we address the problem of developing preconditioners for solving a linear system of the form

$$Ax = b, \tag{1}$$

where A is a general sparse matrix of dimension N. The incomplete LU (ILU) factorization with no fill-in, or ILU(0) [36], is one of the most popular preconditioners currently available. Its implementation on high-performance computers can be optimized by a technique referred to as "level scheduling" or "wavefront ordering"; see, e.g., [3]. A notable disadvantage of ILU(0) is that it is a rather crude approximation, and for this reason it is unreliable when used to solve problems arising from certain applications, such as computational fluid dynamics. To improve the efficiency and robustness of ILU factorizations, many alternatives which allow higher levels of fill-in in the ILU factorizations have been developed [24, 38, 13, 12, 53, 52, 45]. Although these alternatives are more robust than the low-accuracy ILU(0) or symmetric successive overrelaxation (SSOR), they are often intrinsically sequential.

A number of different approaches have been advocated to remedy the "sequential nature" of the preconditioners developed in the 1970s and later; see, for example, the survey papers [14, 42]. The first of these approaches was motivated by vectorization and consisted mainly of replacing occurrences of matrix inverses by polynomials in these matrices. For example, the solutions of bidiagonal systems that arise in the forward and backward solutions on the ILU(0) preconditioning for model problems were replaced by low-degree polynomial expansions in the matrices [47]. The paper [27], and subsequently a few others, advocated using polynomials in A as preconditioners. A second class of methods that has been developed consists of introducing parallelism by exploiting "graph coloring," or *multicoloring*, as we will refer to it here. The unknowns are colored in such a way that no two unknowns of the same color are coupled by an equation. In the simplest case of the five-point matrix arising from the centered difference discretization of the Laplacian in two- or three-dimensional spaces, only two colors are needed, and they are commonly referred to as "red" and "black." When the unknowns of the same color are numbered consecutively, and a standard ILU factorization

*Received by the editors June 1, 1993; accepted for publication (in revised form) March 10, 1994. This research was supported by ARPA under contract number 60NANB2D1272 and by the NSF under grant NSF/CCR-9214116.

†University of Minnesota, Computer Science Department, 4-192 EE/CSci Building, 200 Union Street S.E., Minneapolis, MN 55455.

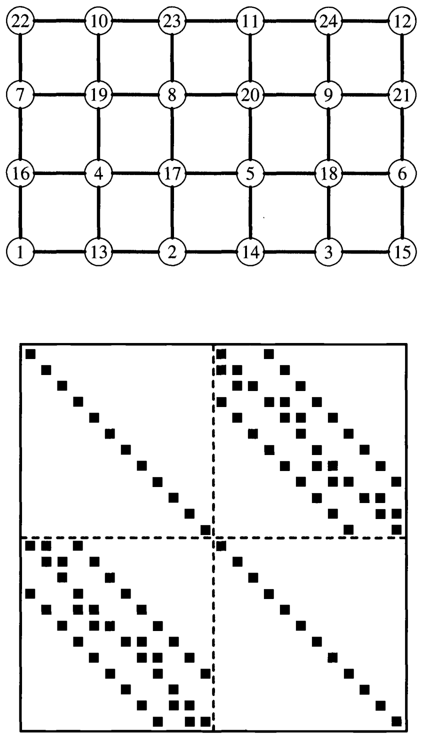
is applied to the reordered system, then a large degree of parallelism is available in both the preprocessing phase and the preconditioning operations, during the iteration phase. A well-known drawback of this approach is that the number of iterations to achieve convergence may increase substantially, when compared with that required for the original system [19, 18, 20]. More generally, preconditioners that allow a large degree of parallelism such as, in the simplest case, diagonal scaling, tend to necessitate a much larger number of iterations to converge when compared with their sequential counterparts, e.g., the standard ILU.

However, experience suggests that a good remedy for regaining an acceptable level of convergence rate is to *increase the accuracy of the underlying preconditioning*. For example, ILU preconditioners with more fill-in, or multistep SSOR or SOR preconditioners, may be used. Experiments in [44] reveal that an approach based on SOR(k) preconditioning, in which each preconditioning operation consists of k steps of SOR sweeps, is superior to the best optimized ILU preconditioning on some problems. Unfortunately, for the standard ILU factorization, the use of a higher level of fill-in destroys the structure obtained from the multicolor reordering.

In this paper we consider a technique which is based on exploiting the idea of successive independent set orderings [33], a simpler form of multicoloring. This technique is very closely related to multifrontal elimination [15, 40], a classical method that is employed in the parallel implementations of sparse direct methods. Both multifrontal elimination and ILUM (ILU with multi-elimination) rely on the fact that at a given stage of Gaussian elimination, there are many rows that can be eliminated simultanously. The set that consists of such rows is called an independent set. The idea then is to find this set, and then eliminate the unknowns associated with it, to obtain a smaller reduced system. This reduction process is applied recursively a few times to the consecutive reduced systems until the system can be solved by a direct solver (multifrontal) or by an iterative technique (ILUM). In ILUM, we perform the reductions approximately, with the help of a standard threshold strategy, in order to control the sparsity of the L and U factors.

We start by describing the ideas of multicoloring and independent set orderings. Then we briefly show how these ideas can be exploited to develop direct solution methods and, finally, we derive incomplete factorization techniques by introducing numerical dropping strategies.

2. Independent set ordering and multicoloring. Graph theory provides numerous useful tools in sparse matrix computations and can be helpful in unraveling parallelism in standard algorithms [25, 15]. We recall that the adjacency graph of a sparse matrix is a graph $G = (V, E)$, whose N vertices in V represent the N unknowns and whose edges represent the binary relations established by the equations in the following manner: there is an edge from node j to node i when $a_{ij} \neq 0$. This edge will therefore represent the binary relation *equation i involves unknown j*, or equivalently *the unknown x_i depends on unknown x_j*. Note that the graph is directed, except when the matrix has a symmetric pattern ($a_{ij} \neq 0$ iff $a_{ji} \neq 0$ for all $1 \leq i, j \leq N$). Parallelism in the Gaussian elimination process can be obtained by finding unknowns that are independent at a given stage of the elimination, i.e., unknowns that do not depend on each other according to the above binary relation. The rows corresponding to such unknowns can then be used as pivots simultaneously. Thus, in one extreme, when the matrix is diagonal, all unknowns are independent. On the other extreme, when a matrix is dense, each unknown will depend on all other unknowns. Sparse matrices lie somewhere in between these two extremes. Multicoloring techniques attempt to find sets of independent equations by coloring the vertices of a graph so that neighboring vertices have different colors. Independent set ordering can be viewed as a less restrictive form of multicoloring, in which a set of vertices in the adjacency graph is found so that no equation in the set involves unknowns



Red-Black ordering

Corresponding Matrix

FIG. 1. *Red-black labeling of a* 6×4 *rectangular mesh and associated matrix.*

from the same set. Multicoloring is best known in the partial differential equation context for a two-dimensional finite difference grid (five-point operator), which is our starting point in this discussion.

2.1. Red-black ordering for finite difference grids. For simple two-dimensional centered finite difference grids, we can easily separate the grid points into two sets, red and black, so that the nodes in one set are adjacent only to nodes from the other set. This two-color (red-black) ordering is illustrated in Figure 1 for a 6×4 grid.

If we reorder the unknowns in such a way as to list the red unknowns first together and then the black ones, as is illustrated in Figure 1, we will obtain a system of the form

$$\begin{pmatrix} D & F \\ E & C \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}, \tag{2}$$

where D and C are diagonal matrices. Matrices that can be permuted into the above form are said to have property A [51]. Several techniques have traditionally been used to exploit the above convenient structure (see, e.g., [23, 41, 21]).

Perhaps the simplest of these approaches is to use a conjugate gradient-type technique combined with the standard ILU(0) preconditioner on the block system (2). Here, the degree of parallelism, i.e., the maximum number of arithmetic operations which can be executed in parallel, is of order N. A drawback is that often the number of iterations is higher than with the natural ordering, but the approach may still be competitive for problems for which the natural ordering requires a relatively small number of steps, e.g., less than 30, to converge.

We have observed [44] that the number of iterations can be reduced back to a competitive level by using a more accurate ILU factorization on the red-black system, e.g., ILU with threshold (ILUT) [45]. In fact the situation may be improved in that for the same level of fill-in p, the red-black ordering will outperform the natural ordering preconditioner for p large enough, in terms of number of iterations required for convergence. However, the fill-in introduced in the "black" part of the system with such high-level ILUs ruin the degree of parallelism achieved from the red-black ordering. A remedy is to resort to similar *high-level SOR or SSOR preconditioners* instead of ILUT [44]. These consist of performing several SOR or SSOR steps in each preconditioning operation, instead of just one, as is traditionally done. A significant advantage of relaxation-type preconditioners is that these higher-level preconditioners do not lose their degree of parallelism as the accuracy increases.

A second method that has been used in conjunction with the red-black ordering is to eliminate the red unknowns by forming the reduced system which involves only the black unknowns, namely,

$$(C - ED^{-1}F)y = g - ED^{-1}f.$$

This linear system is again sparse and has about half as many unknowns. It has been observed that for easy problems, i.e., problems for which the natural ordering requires a relatively small number of steps to converge, this reduced system can often be efficiently solved with only diagonal preconditioning. The preprocessing to compute the reduced system is highly parallel and inexpensive. In addition, the reduced system is usually well conditioned and has some interesting properties when the original system is highly nonsymmetric [21].

A general sparse matrix rarely has property A, and as a result the application of the red-black ordering is quite limited. Fortunately, many of the above techniques can be generalized by using less restrictive forms of reorderings. For example, to exploit the reduced system approach, all we need is to reorder the original matrix into a matrix of the form

$$\begin{pmatrix} D & F \\ E & C \end{pmatrix}, \tag{3}$$

where D is diagonal but C can be arbitrary. There are three ways of generalizing the standard red-black ordering.

1. Independent set orderings, which lead to the form (3) above.
2. Multicolor orderings, which lead to block matrices in which the diagonal blocks are pointwise diagonal.
3. Full-block versions of the above techniques which allow the diagonal matrices to be block diagonal instead where each block is dense.

In the remainder of this paper we will consider the first and second generalizations. The block versions arise naturally in the solution of partial differential equations (PDEs) when each mesh point involves several unknowns. We also note that a further generalization would consist of allowing these diagonal blocks to be sparse matrices.

2.2. Independent set orderings. We now consider generalizations of the red-black ordering which consist of transforming the matrix into the block form (3), where D is diagonal. Let $G = (V, E)$ denote the adjacency graph of the matrix, and let (x, y) denote an edge from vertex x to vertex y. An *independent set* S is a subset of the vertex set V such that

$$\text{if } x \in S \quad \text{then} \quad \{\, (x, y) \in E \ \text{ or } \ (y, x) \in E \,\} \rightarrow y \notin S.$$

In words, elements of S are not allowed to be connected to other elements of S either by incoming or outgoing edges. An *independent set* is maximal if it cannot be augmented by elements in its complement to form a larger independent set. Note that a maximal independent set is by no means the largest possible independent set that can be found. In fact, finding the independent set of maximum cardinality is NP-hard [32]. Independent set orderings have been used mainly in the context of parallel direct solution techniques for sparse matrices [33, 34, 11], and multifrontal techniques [15] can be viewed as a particular case. One of the goals of this paper is to show how to exploit these ideas in the context of iterative methods.

There are a number of simple and inexpensive heuristics for finding large maximal independent sets. For our purposes simple greedy algorithms such as the ones to be described next are inexpensive and yield enough parallelism in general. In the following we will use the term *independent set* to always refer to *maximal independent set.*

A simple greedy procedure for finding an independent set S is to traverse the graph and accept each visited node as a new member of S, if it is not already marked. We then mark this new member and all of its nearest neighbors. Note that by nearest neighbors of a node x we mean the adjacent nodes, linked to x by incoming or outgoing edges.

ALGORITHM 2.1. Greedy algorithm for independent set ordering.

```
Let $S = \emptyset$.
For $j = 1, 2, \ldots, n$ Do:
        If node $j$ is not marked then
                $S = S \cup \{j\}$.
                Mark $j$ and all its nearest neighbors.
        endif
enddo
```

In the above algorithm, the nodes are visited in the natural order $1, 2, \ldots, n$ but we can also visit them in any permutation $\{i_1, \ldots, i_n\}$ of $\{1, 2, \ldots, n\}$. In what follows we denote by $|S|$ the size of S, i.e., its cardinal. Since the size of the reduced system is $n - |S|$ it is reasonable to attempt to maximize the size of S in order to obtain a small reduced system, although the size is not all that matters.

We would like to give a rough idea of the size of S. Recall that the degree of a vertex v is the total number of edges that are adjacent to v. Assume that the maximum degree of each node does not exceed ν. Whenever the above algorithm accepts a node as a new member of S it potentially puts all its nearest neighbors, i.e., at most ν nodes, in the complement of S. Therefore, the size $n - |S|$ of its complement is such that $n - |S| \leq \nu |S|$ and as a result,

$$|S| \geq \frac{n}{1+\nu}.$$

This lower bound can be improved slightly by observing that we can replace ν by the maximum degree ν_S of *all the vertices that constitute* S, resulting in the inequality

$$|S| \geq \frac{n}{1+\nu_S},$$

which suggests that it may be a good idea to visit first the nodes with smaller degrees. In fact this observation leads to a general heuristic regarding a good order of traversal. We can view the algorithm as follows. Each time a node is visited, we remove it and its nearest neighbors from the graph and then visit a node from the remaining graph and continue in the same manner until exhaustion of all nodes. Every node that is thus visited is a member of S, and its nearest neighbors are members of the complement $\bar{S}$ of the set S. As a result, if we call ν_i the degree of the node visited at step i, adjusted for all the edge deletions resulting from the previous visitation steps, then the number n_i of nodes that are left at step i satisfies the relation

$$n_i = n_{i-1} - \nu_i - 1.$$

The process adds a new element to the set S at each step and stops when $n_i = 0$. In order to maximize $|S|$ we need to maximize the number of steps in the procedure. The difficulty in the analysis comes from the fact that the degrees are updated at each step i because of the removal of the edges associated with the removed nodes. All we can say is that if we wish to lengthen the process, a rule of thumb would be to visit the nodes that have the smallest degrees first.

ALGORITHM 2.2. Independent set ordering with increasing degree traversal.

```
Let $S = \emptyset$. Find an ordering $i_1, \dots, i_n$ of the nodes by increasing degree.
For $j = 1, 2, \dots n$ Do:
    If node $i_j$ is not marked then
        $S = S \cup \{i_j\}$.
        Mark $i_j$ and all its nearest neighbors.
    endif
enddo
```

A refinement to the above algorithm would be to update the degrees of all nodes involved in a removal and dynamically select the one with smallest degree as the next node to be visited. This can be efficiently implemented using a min-heap data structure [9]. A different heuristic is to attempt to maximize the number of elements in S by a form of local optimization that determines the order of traversal dynamically. In the following the action of removing a vertex from a graph consists of deleting the vertex and all edges incident to/from this vertex.

ALGORITHM 2.3. Locally optimal algorithm for independent set ordering.

```
Set $S = \emptyset$ and $n_{left} = n$.
While $n_{left} > 0$ do
        Select the vertex with minimum degree in current graph.
        Add this vertex to $S$, then
        Remove it and all of its nearest neighbors from graph
        Update degrees of all vertices
        $n_{left} := n_{left} -$ number_of_removed_vertices.
endwhile
```

There is a striking similarity with the minimal-degree ordering algorithm used in sparse Gaussian elimination. The only difference is that the elimination of a node does not introduce what is referred to as fill-in in Gaussian elimination, i.e., the edges of the removed nodes are simply deleted.

There are other similar techniques for generating independent sets. The method described in [33] for an alternative ordering for Gaussian elimination starts with the observation that constructing a large independent set is equivalent to building a small complement $\bar{S}$ to S. One observes that all the edges of the graph are edges between vertices in $\bar{S}$, i.e., the vertices in $\bar{S}$ form a *vertex cover* of the graph. To find a small vertex cover, Leuze suggests a locally optimal technique which can be viewed as the dual of the previous algorithm. At each step of the procedure, the vertex of *maximum* degree is found, and this vertex together with all the incident edges are removed. The process is continued until exhaustion of all edges.

ALGORITHM 2.4. Vertex cover algorithm.

```
Set $\bar{S} = \emptyset$
While (there are still edges left in $G$) Do
```

TABLE 1
Performance of four different independent set ordering algorithms for first test problem.

	First reduction		Second reduction	
Method	N_1	NZ_1	N_2	NZ_2
Alg. 2.1	514	10800	460	15920
Alg. 2.2	523	10414	462	13842
Alg. 2.3	511	10826	454	14868
Alg. 2.4	530	10260	469	13386

TABLE 2
Performance of four different independent set ordering algorithms for matrix JPWH_991.

	First reduction		Second reduction	
Method	N_1	NZ_1	N_2	NZ_2
With tol = 0.0				
Alg. 2.1	630	7902	512	12820
Alg. 2.2	603	5640	439	7084
Alg. 2.3	583	6121	433	8381
Alg. 2.4	594	5995	438	7783
With tol = 0.05				
Alg. 2.1	514	8183	455	6481
Alg. 2.2	523	7820	462	6185
Alg. 2.3	511	8209	452	6513
Alg. 2.4	530	7675	465	6734

```
        Select the vertex with maximum degree in current graph.
        Add this vertex to $\bar{S}$, then
        remove it and all its adjacent edges from the graph.
    EndWhile
```

One drawback with this type of "local minimization" technique illustrated by the last two algorithms is its cost. The fact that the improvements over the simpler procedures (Algorithms 2.1 and 2.2) are likely to be small suggests that it is probably preferable to use these less expensive algorithms in practice. This will be confirmed by our experiments in §4.

We now illustrate these algorithms with the help of a little comparison. We consider two test matrices. The first is a 9-point matrix obtained simply by squaring the 5-point discretization of the Laplacian on a 25×25 grid. This matrix of size $N = 625$ has 7629 nonzero elements and it pattern is symmetric. The second matrix is taken from the Harwell-Boeing collection and is called JPWH_991; see [16, 17]. It is of dimension $N = 991$ and has a total of 6,027 nonzero elements, and its pattern is nonsymmetric.

Tables 1 and 2 show the results for two reduction steps (see §3). In addition, Table 2 illustrates a simple dropping strategy by showing the effect of adding a drop tolerance, called tol in the table. When forming the reduced system, a row a_i will be modified by linear combinations of rows corresponding to adjacent nodes in the graph. If $\|a_i\|$ is the 2-norm of a_i, then every element obtained in this transformation is dropped if its magnitude is less than $tol \times \|a_i\|$. In the table N_i is the size of the reduced system obtained at the ith level reduction, and NZ_i its number of nonzero elements. As can be seen from the tables the difference in the performances is rather small. An interesting observation is that Algorithm 2.3 seems to be best at minimizing the size of the reduced system, i.e., maximizing the size of S, whereas Algorithm 2.4 seems to be better at reducing the amount of fill-in generated in the reduced system. This is somewhat expected because of the nature of the heuristics used. Overall, Algorithm 2.2 seems to perform remarkably well given the simplicity of the underlying heuristic.

2.3. Multicoloring for arbitrary sparse matrices. Multicoloring techniques have often been used in the context of PDEs as a means of introducing parallelism [39, 37, 1, 30, 31, 28]. The goal here is to color the vertices of an arbitrary adjacency graph in such a way that two adjacent nodes do not have a common color. In terms of graphs, this means that we must find a partition $S_1, S_2, \ldots, S_k$ of the vertex set V, such that

$$\text{if } (x, y) \in E \text{ with } x \in S_i \text{ and } y \in S_j \quad \text{then } i \neq j.$$

Clearly, the red-black ordering is just a particular case with $k = 2$.

As for independent set orderings, finding a multicolor ordering is inexpensive, provided we do not seek to achieve optimality. Multicoloring ideas have been employed in particular to understand the theory of relaxation techniques [51, 49] as well as for deriving efficient alternative formulations of some relaxation algorithms [49, 23]. More recently, they have emerged as useful tools for introducing parallelism in iterative methods; see, for example, [2, 1, 39, 20, 37]. Multicoloring is also commonly employed in a slightly different form (coloring elements (or edges) as opposed to nodes) in finite elements (or finite volume) techniques especially when using the element-by-element approach [6, 50, 26, 22, 46]. Note that "multicoloring" is generally based on a graph coloring of some sort, and that there are numerous other applications of these techniques which are unrelated to parallel computing. Thus, in a quite different context, a form of multicoloring has also been used in [7] to extract independent columns in a sparse matrix for the purpose of numerically evaluating Jacobians with difference formulas.

There is a rich literature on graph coloring, and we refer to [7, 28] for references and a few algorithms that attempt to achieve optimality by some heuristics similar to the ones introduced earlier for independent set orderings. In this paper we will only consider a simple greedy technique for obtaining a multicoloring of an arbitrary graph. Initially, the algorithm assigns a color of zero to each node i. Then, it traverses the graph in the natural order and assigns the smallest positive *admissible* color to each node i visited. Here, an admissible color is a color not already assigned to any of the neighbors of node i. In the following description of the greedy algorithm, we use the notation Adj(i) to represent the set of nodes that are adjacent to node i.

ALGORITHM 2.5. Greedy multicoloring algorithm.

```
Set $Color(i) = 0, i = 1, \ldots, N$.
For $i = 1, 2, \ldots, N$ Do:
    $\text{Color}(i) = \min\{k > 0 \mid k \neq \text{Color}(j), \forall\, j \in \text{Adj}(i)\}$.
EndDo
```

At the end of the algorithm, each node i will be assigned the color $Color(i)$. This procedure is illustrated in Figure 2.

In a manner similar to that for independent set orderings we can traverse the nodes in any order $\{i_1, i_2, \ldots, i_n\}$, and this initial order of traversal may be important in reducing the number of colors. However, the difference between a poor ordering and a good one is usually small. Here are a few additional properties concerning this greedy algorithm.

- If a graph is bipartite (i.e., two-colorable), the algorithm will find the optimal two-color (red-black) ordering for *breadth-first search* traversals.
- Any chain traversal, i.e., a traversal following a *path through all the nodes* in the graph, will also find the optimal two-color ordering for any bipartite graph.
- The number of colors needed does not exceed $\nu + 1$ where ν is the maximum degree of all nodes.

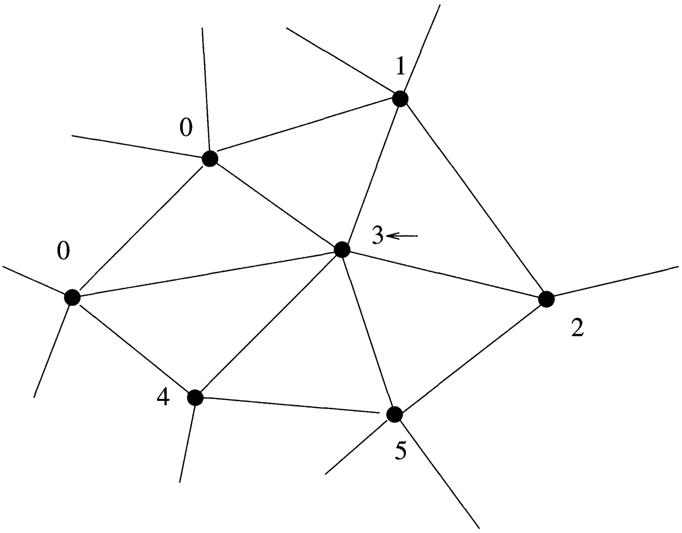


FIG. 2. *The greedy multicoloring algorithm. The node being colored is indicated by an arrow. It will be assigned color number* 3, *the smallest positive integer different from* 1, 2, 4, 5.

Here, we recall that a breadth-first search traversal starts from an arbitrary node and visits its nearest neighbors which form the first level. Then it visits the nearest neighbors of all the nodes in level 1, which have not been visited. These will constitute the second level. The algorithm continues in this fashion until all nodes have been visited. The proofs of the above properties are straightforward and are omitted. In [7] it is proved that a different ordering, referred to as the "incidence degree" ordering, also achieves optimality for bipartite graphs.

For the natural traversal ordering, we can parallelize this graph coloring algorithm in one of several ways. A simple mechanism implemented in [35] is to impose a local order in which the coloring is to be done. For example, a given node can wait until all its nearest neighbors whose processor numbers are less than its own have colored their nodes. Once this is done the processor assigns colors to each of its nodes and then sends the color information needed by the nearest neighbors with higher processor number. Several other alternative algorithms exist; see, for example, [29].

3. ILUM: a multi-elimination ILU factorization.

A parallel direct solver based on performing several successive levels of independent set orderings and reduction was suggested in [33] and in a slightly more general form in [10]. Although the ideas were presented differently, their essence is that when we eliminate the unknowns associated with an independent set we obtain another system, which is a smaller sparse linear system. We can then find an independent set for this reduced system and repeat the process of reduction. We refer to the resulting second reduced system as the second-level reduced system. The process can be repeated recursively a few times. As the level of the reduction increases, the reduced systems gradually lose their sparsity. A direct solution method would consist of continuing the reduction until the reduced system is small enough or dense enough that we can resort to a dense Gaussian elimination to solve it. A sparse direct solution method based on a similar argument was suggested in [33]. In [44] we suggested performing a small number of reduction steps and then switching to a traditional iterative solver, preferably one that has a high level of parallelism such as multicolor successive overrelaxation (SOR), accelerated with GMRES.

ILUM is a form of block preconditioning [8, 4] in which reordering is used to ensure that the diagonal blocks to be inverted during the factorization remain diagonal matrices.

After a brief review of the direct solution method based on independent set ordering, we will show how to exploit this approach to derive ILU factorization strategies based on a drop tolerance.

3.1. Multi-level reduced systems. We start by a brief discussion of an *exact* reduction step. Let A_j be the matrix obtained at the jth step of the reduction, $j = 0, \ldots, nlev$ with $A_0 = A$. Assume that an independent set ordering is applied to A_j and that the matrix is permuted accordingly as follows:

$$P_j A_j P_j^T = \begin{pmatrix} D_j & F_j \\ E_j & C_j \end{pmatrix}, \tag{4}$$

where D_j is a diagonal matrix. We can now reduce the system by eliminating the unknowns of the independent set to get the next reduced matrix via the formula

$$A_{j+1} = C_j - E_j D_j^{-1} F_j. \tag{5}$$

Note that we have implicitly performed the block LU factorization

$$P_j A_j P_j^T = \begin{pmatrix} D_j & F_j \\ E_j & C_j \end{pmatrix} = \begin{pmatrix} I & 0 \\ E_j D_j^{-1} & I \end{pmatrix} \times \begin{pmatrix} D_j & F_j \\ 0 & A_{j+1} \end{pmatrix} \tag{6}$$

with A_{j+1} defined above. As a result, in order to solve a system with the matrix A_j we need to perform a forward and backward substitution with the block matrices on the right-hand side of (6). The backward solution involves solving a system with the matrix A_{j+1} and the forward solution can be performed with a diagonal scaling and a matrix-vector product.

We can use this block factorization approach recursively until we obtain a system that is small enough to be solved with a standard method, e.g., a dense Gaussian elimination. We must save the transformations used in the elimination process, i.e., the matrices $E_j D_j^{-1}$ and the matrices F_j. The permutation matrices P_j can also be saved, or we can explicitly permute the matrices involved in the factorization at each new reordering step.

3.2. ILUM. Clearly, the successive reduced systems obtained in the way described above will become more and more expensive to form and store as the number of levels increases. This is due to the fill-in introduced by the elimination process. A common cure for this in developing preconditioners is to neglect some of the fill-in introduced by using a simple dropping strategy as we form the reduced systems. For example, we can drop any fill-in element introduced, whenever its size is less than a given tolerance τ times the 2-norm of the original row. Thus, an "approximate" version of the successive reduction steps can be used to provide an approximate solution $M^{-1}v$ to $A^{-1}v$ for any given v. This can be used to precondition the original linear system. Conceptually, the modification leading to an "incomplete" factorization consists of replacing (5) by

$$A_{j+1} = (C_j - E_j D_j^{-1} F_j) - R_j, \tag{7}$$

in which R_j is the matrix of the elements that are dropped in this reduction step. Globally, the algorithm can be viewed as a form of block incomplete LU [8, 5], with permutations.

Thus, we have a succession of incomplete block LU factorizations of the form

$$P_j A_j P_j^T = \begin{pmatrix} D_j & F_j \\ E_j & C_j \end{pmatrix} = \begin{pmatrix} I & 0 \\ E_j D_j^{-1} & I \end{pmatrix} \times \begin{pmatrix} D_j & F_j \\ 0 & A_{j+1} \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & R_j \end{pmatrix} \tag{8}$$

with A_{j+1} defined by (7). We can now recursively find an independent set ordering for the new matrix A_{j+1} and reduce it again in the same manner. Note that we need not save the successive A_j matrices, but only the last one that is generated. We must also save the sequence of sparse matrices

$$B_{j+1} = \begin{pmatrix} D_j & F_j \\ E_j D_j^{-1} & 0 \end{pmatrix}, \tag{9}$$

which contain the transformation needed at level j of the reduction. We can discard the successive permutation matrices P_j if we apply them to the previous B_i matrices as soon as these permutation matrices are known. We then need only the global permutation which is the product of all these successive permutations. The successive B_j matrices without the diagonal D_j are stored as a succession of sparse matrices in a sparse row format. The diagonals D_j are all stored in an N-dimensional array.

An illustration of the matrices obtained after three reduction steps is shown in Figure 3. The algorithm used for the independent set ordering in the illustration is Algorithm 2.3. The original matrix is a five-point matrix associated with a 15×15 grid, and is therefore of size $N = 225$. Here the matrices B_1, B_2, B_3 (with permutations applied) are shown together with the matrix A_3 occupying the location of the 0 block in (9).

We refer to this incomplete factorization as ILUM (ILU with multi-elimination). The preprocessing phase consists of a succession of $nlev$ applications of the three steps: (1) finding the independent set ordering, (2) permuting the matrix, and (3) reducing it.

ALGORITHM 3.1. ILUM: preprocessing phase.

```
Set $A_0 = A$.
For $j = 0, 1, \ldots, nlev - 1$ Do:
        Find an independent set ordering permutation $P_j$ for $A_j$;
        Apply $P_j$ to $A_j$ to permute it into the form (4);
        Apply $P_j$ to $B_1, \ldots, B_j$;
        Apply $P_j$ to $P_0, \ldots, P_{j-1}$;
        Compute the matrices $A_{j+1}$ and $B_{j+1}$ defined by (7) and (9).
Enddo
```

In the backward and forward solution phases we must solve the last reduced system but we need not solve it accurately. We can, for example, solve it according to the level of tolerance that we have allowed in the dropping strategy during the preprocessing phase. Observe that since we would like to solve the linear system inaccurately, we should only use an accelerator that allows variations in the preconditioning. Such methods have been developed in [43] and [48]. Alternatively, we can use a fixed number of multicolor SOR or SSOR steps. The implementation of the ILUM preconditioner corresponding to this strategy is rather complicated and involves several parameters.

In order to describe the forward and backward solution we need to introduce some notation. We start by applying the "global permutation," i.e., the product $P_{nlev-1}P_{nlev-2}\ldots P_0$, to the right-hand side. We overwrite the result on the current solution vector, an N-vector which we call x_0. We now partition this vector into

$$x_0 = \begin{pmatrix} y_0 \\ x_1 \end{pmatrix}$$

according to the partitioning (4). The forward step consists of transforming the second component of the right-hand side as

$$x_1 := x_1 - E_0 D_0^{-1} y_0.$$

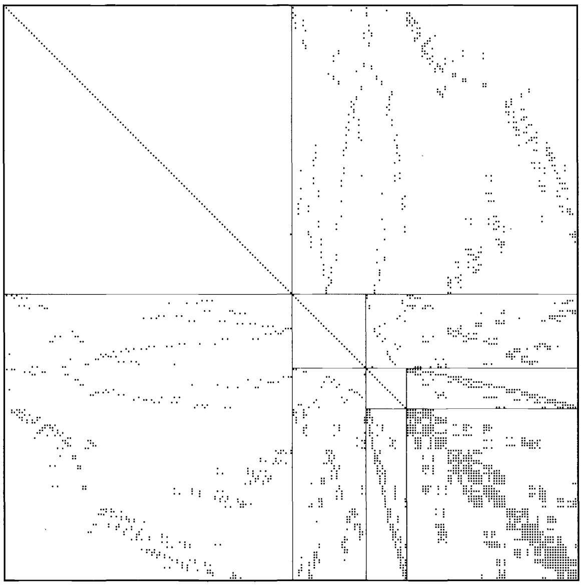


FIG. 3. *Illustration of the processed matrices obtained from three steps of independent set ordering and reductions.*

Now x_1 is partitioned in the same manner as x_0 and the forward elimination is continued the same way. Thus, at each step we partition each x_j as

$$x_j = \begin{pmatrix} y_j \\ x_{j+1} \end{pmatrix}.$$

A forward elimination step defines the new x_{j+1} using the old x_{j+1} and y_j for $j = 0, \ldots, nlev - 1$ while a backward step defines y_j using the old y_j and x_{j+1}, for $j = nlev - 1, \ldots, 0$. Algorithm 3.2 describes the general structure of the forward and backward solution sweeps. Because we apply the global permutation at the beginning, we need not apply the successive permutations. However, we need to permute the final result obtained back into the original ordering.

ALGORITHM 3.2. ILUM: Forward and backward solutions.

```
1. Apply global permutation to right-hand side $b$ and copy into $x_0$.
2. For $j = 0, 1, \ldots, nlev - 1$ do: [Forward sweep]
       $x_{j+1} := x_{j+1} - E_j D_j^{-1} y_j$
   EndDo
3. Solve with a relative tolerance $\epsilon$:
       $A_{nlev} x_{nlev} := x_{nlev}$.
4. For $j = nlev - 1, \ldots, 1, 0$ do: [Backward solution]
       $y_j := D_j^{-1}(y_j - F_j x_{j+1})$.
   EndDo
5. Permute the resulting solution vector back to the original ordering
   to obtain the solution $x$.
```

A major source of difficulty with the use of ILUM lies in its implementation. The implementation issues are similar to those that arise when implementing parallel direct solution methods for sparse linear systems. Some of the issues have been briefly discussed above. Here are some of the important questions that arise, along with some comments or solutions.

1. Should we permute the matrices A_j explicitly or is it preferable to avoid step 3 in Algorithm 3.1? There are two additional burdens if we do not permute the matrices. First, we must store the successive permutations and second, we must apply the permutations during the forward and backward solutions when passing from one level to the other. As was mentioned, in our implementation we explicitly permute the matrices.
2. How to store the successive matrices B_j? In our implementation these are stored in sequence one after the other in a single data structure. As a result the data structure used for storing the successive matrices has two levels of pointers. It is also possible to store all the B_j matrices as a single sparse matrix, the upper part in row format and the lower part in column format. We would then need two sparse data structures and an additional pointer.
3. What number of levels should we use? How do we relate the number of levels to the tolerance used in the factorization? These are difficult questions to answer in a rigorous way. The number of levels is left as a parameter in our implementation.

4. Numerical tests. In this section we provide a few experimental results to (i) help give an idea of the performance of the ILUM preconditioner when compared with similar techniques and (ii) examine how these performances vary with respect to the type of independent set ordering used. All experiments have been performed on a Cray-2 in single precision (64-bit arithmetic) and the times are in seconds. Although parallelism is not exploited our implementation of the iterative phase, which excludes the preprocessing required to compute the ILU factorization itself, takes advantage of vectorization.

In all tests, we construct the right-hand sides artificially to be of the form $b = Ae$, where the solution e is the vector whose components are all ones. The initial guess is taken to be a random vector. Although some of the test matrices used in these experiments are associated with regular grids, this was not exploited, i.e., the matrices are considered as general sparse. The iterations were stopped as soon as the 2-norm of the residual was reduced by a factor of $\epsilon = 10^{-7}$.

4.1. An elliptic problem. We consider the partial differential equation

$$-\Delta u + \gamma \left(\frac{\partial e^{xy} u}{\partial x} + \frac{\partial e^{-xy} u}{\partial y} \right) + \alpha u = f,$$

with Dirichlet boundary conditions and $\gamma = 10, \alpha = -60$, discretized with centered differences on a $27 \times 27 \times 27$ grid. This leads to a linear system with $N = 25^3 = 15,625$ unknowns. We refer to the linear system associated with this matrix as Problem 1.

The results obtained with a drop tolerance of $\tau = 0.0001$ in ILUM are shown in Table 3. The last reduced system is solved with GMRES(20) preconditioned by SOR(1) and using a tolerance of $\epsilon = 0.01$ for the stopping criterion. Note that in all the SOR preconditioning operations we use $\omega = 1$ as a relaxation parameter. The reordering algorithms tested are indicated on the left column, in which "Multicolor" refers to taking the first set (color) obtained from the greedy multicoloring Algorithm 2.5. The column "its_T" shows the time required to solve the system excluding the preprocessing phase required to compute the incomplete factorization. The total time, including preprocessing, is shown in the column "tot_T." The column "memory" shows the total memory requirement in thousands of words to store the real values (excluding integer indices) of the incomplete factorization. The numbers "its-out" refer to the number of outer iterations required, i.e., the number of ILUM preconditioned GMRES steps needed to achieve convergence. The numbers "its-tot" refer to the total number

TABLE 3

Performance of GMRES(10)*-ILUM preconditioning for Problem* 1, *using different independent set ordering algorithms.*

Method		Performance				
Algorithm	*nlev*	tot_T	its_T	Memory	its-out	its-tot
Multicolor	1	3.52	1.54	7.9	4	56
	2	7.30	3.69	142.6	4	52
	3	11.41	5.58	166.5	5	65
Alg. 2.1	1	3.26	1.49	7.9	4	56
	2	7.41	4.00	142.9	4	55
	3	10.94	5.64	173.5	5	68
Alg. 2.2	1	3.91	1.78	9.3	4	60
	2	7.56	3.77	127.4	4	55
	3	11.29	5.61	161.1	5	68
Alg. 2.3	1	4.46	1.52	7.9	4	57
	2	8.95	3.59	143.9	4	51
	3	14.04	5.56	176.4	5	68
Alg. 2.4	1	4.05	1.59	5.4	4	55
	2	8.28	3.53	138.1	4	51
	3	14.00	6.17	165.1	5	67

of inner iterations required, i.e., the overall total number of matrix-vector products in the calls to GMRES(20)-SOR(1) needed to solve all the reduced systems occurring throughout the solution of the linear system under consideration.

4.2. Experiments with Harwell-Boeing matrices. We first consider a linear system made up from the matrix Sherman-3 of the Harwell-Boeing collection [16, 17]. This matrix is of dimension $N = 5,005$ and has $nz = 20,033$ nonzero elements. It arises in an IMPES (IMplicit Pressure, Explicit Saturation) simulation of a black-oil reservoir on a $35 \times 11 \times 13$ grid. We refer to the linear system associated with this matrix as Problem 2. Results similar to those of the previous example are shown in Table 4.

The number of outer iterations is roughly the same in all cases but the number of inner iterations varies rather substantially. We should point out that the preprocessing to build the incomplete factorization has not been optimized. It is possible to improve performance by exploiting parallelism in the elimination since the main operation in forming the reduced systems consists of a sparse matrix-matrix product. In addition, even if the preprocessing costs remain high, this technique may be appealing for solving linear systems with several right-hand sides on parallel or vector computers. It is worth noting that if we ignore the preprocessing times then for Problem 2, some of the best performances, in terms of execution time, are obtained with larger numbers of reductions, a situation which is opposite to that of Problem 1. This depends largely on the parameters used in the factorization.

For comparison, we show in Table 5 typical execution times using an optimized ILUT preconditioned GMRES approach. The ILUT(p, τ) preconditioner described in [45] is a dual-threshold-based incomplete LU factorization which performs numerical dropping based on a relative tolerance τ and which retains at most the p largest fill-in elements in L and in U. The optimization of ILUT on the CRAY consists of using level scheduling [3] as well as jagged diagonal data structures to optimize matrix-vector products. Note that if we ignore preprocessing times, the best times achieved with ILUT and ILUM are comparable, but the degree of parallelism in ILUM is much higher that in ILUT.

Finally, we show the performance of ILUM and an optimized ILUT preconditioned GMRES approach, both with various parameters, on eight matrices from the Harwell-Boeing collection. The two methods are not comparable for similar values of their parameters. How-

TABLE 4

Performance of GMRES(10)*-ILUM preconditioning for matrix Sherman* 3, *using different independent set ordering algorithms.*

Method		Performance				
Algorithm	*nlev*	tot_T	its_T	Memory	its-out	its-tot
Multicolor	1	1.44	1.17	19.4	6	232
	2	2.68	2.21	25.6	7	194
	3	2.94	2.25	30.3	8	171
	10	3.64	1.56	53.3	11	90
	20	4.71	1.03	71.8	10	43
Alg-2.1	1	1.57	1.33	19.5	6	261
	2	2.70	2.27	25.4	7	185
	3	2.73	2.13	30.3	8	167
	10	3.22	1.31	55.9	10	73
	20	4.41	1.03	75.2	11	39
Alg-2.2	1	2.54	2.22	15.7	7	394
	2	3.70	3.16	22.3	7	318
	3	2.90	2.16	27.9	9	190
	10	3.14	1.09	51.8	9	68
	20	4.43	0.89	70.5	9	35
Alg-2.3	1	2.06	1.60	20.5	7	329
	2	3.25	2.52	25.7	8	232
	3	3.36	2.29	30.5	8	190
	10	4.13	1.14	54.2	10	67
	20	5.762	0.76	72.4	9	31
Alg-2.4	1	2.17	1.84	19.7	7	363
	2	3.37	2.74	24.9	7	241
	3	3.69	2.73	29.6	8	218
	10	4.47	1.24	52.3	9	78
	20	6.49	0.80	70.3	8	33

TABLE 5

Performance of GMRES(10)*-ILUT(p,τ), with* $p = 10$ *and* $\tau = 0.0001$ *using level scheduling for the triangular system solutions.*

	ILUT time	GMRES time	tot. time	its
Problem 1	12.1	2.70	14.8	25
Problem 2	1.69	0.761	2.45	20

ever, the results will give an indication of how the two methods may compare for reasonable choices of the parameters and for unstructured matrices. The sizes (N) of these matrices and their number of nonzero elements (Nz) are shown in the first column of Table 6. Of these matrices, only ORSREG1, SHERMAN1, and SHERMAN5 are regularly structured.

Here, the ILUT(p, τ) preconditioner was used with $\tau = 10^{-4}$, and the level-of-fill parameter p takes the values 0, 5, 10, 15. For ILUM we took the same value for τ and retained at most 20 elements in each row of the reduced system. Since the size of the reduced matrices decreases at each level, it is difficult to compare the number of nonzero elements obtained with a given value of p for ILUM and ILUT. The reduced systems are solved with GMRES(10) with diagonal preconditioning. Only one outer GMRES(10) iteration is performed for each different system. The algorithm used for obtaining the independent sets was Algorithm 2.4. The results are shown in Table 6. In most cases the best iteration times with ILUM are achieved for larger values of the level number. In addition, these times are often better than the level-scheduling implementation of ILUT. The preprocessing times are not optimized for both algorithms, so it is difficult to give a comparison with the current implementations of

TABLE 6
Comparison of ILUM and ILUT on a few matrices from the Harwell-Boeing collection.

	ILUM				ILUT			
Matrix	*nlev*	tot_T	its_T	its	*p*	tot_T	its_T	its
FS7602	2	1.84	1.73	85	0	0.58	0.47	95
$N = 760$	8	0.43	0.17	12	5	0.27	0.10	17
$Nz = 5976$	14	0.40	0.06	6	10	0.29	0.06	9
	20	0.45	0.05	5	15	0.33	0.07	9
ORSIRR1	2	0.66	0.53	26	0	0.19	0.08	15
N= 1030	8	0.41	0.13	9	5	0.18	0.06	8
Nz= 6858	14	0.45	0.09	7	10	0.18	0.05	7
	20	0.50	0.09	7	15	0.18	0.06	7
ORSIRR2	2	0.59	0.48	28	0	0.17	0.07	15
N= 886	8	0.38	0.14	11	5	0.17	0.05	8
Nz= 5970	14	0.39	0.08	7	10	0.17	0.05	7
	20	0.44	0.08	7	15	0.17	0.05	7
ORSREG1	2	0.91	0.62	16	0	0.42	0.17	16
N= 2205	8	0.78	0.19	7	5	0.36	0.08	6
Nz= 14133	14	0.89	0.18	7	10	0.35	0.08	6
	20	1.03	0.18	7	15	0.35	0.08	6
PORES2	2	4.08	3.93	200	0	1.17	1.06	171
N= 1224	8	1.19	0.92	59	5	0.62	0.47	47
Nz= 9613	14	1.11	0.77	52	10	0.55	0.39	33
	20	1.14	0.73	51	15	0.52	0.36	27
PORES3	2	0.54	0.48	46	0	0.12	0.07	22
N= 532	8	0.20	0.06	8	5	0.14	0.05	10
Nz=3474	14	0.21	0.04	6	10	0.18	0.05	8
	20	0.24	0.04	6	15	0.22	0.06	7
SHERMAN1	2	0.21	0.12	7	0	0.22	0.15	31
N=1000	8	0.34	0.07	4	5	0.24	0.09	8
Nz= 3750	14	0.48	0.06	4	10	0.32	0.11	6
	20	0.57	0.05	4	15	0.38	0.15	6
SHERMAN5	2	1.24	0.85	15	0	1.25	0.74	45
N=3312	8	1.62	0.61	11	5	1.40	0.44	19
Nz=20793	14	1.86	0.33	7	10	1.81	0.46	15
	20	2.17	0.25	6	15	2.22	0.52	14

the overall process. These experiments do indicate, however, that a good implementation of ILUM may be a competitive approach to vector supercomputers.

5. Conclusion. The ideas of graph coloring and independent set ordering can be exploited to derive highly parallel incomplete LU factorizations. We have developed such incomplete factorizations and tested them on a Cray computer, exploiting only vectorization. However, the implementation of these techniques on massively parallel computers is likely to be complex. In most cases a good compromise may well be to perform a small number of ILUM reductions, perhaps one or two, and then solve the last reduced system with a multicolor multistep SOR preconditioned Krylov subspace iteration [44]. Nevertheless, the general idea of independent set orderings is powerful and can certainly be exploited in many other ways than those described in this paper.

Acknowledgments. The author would like to thank the anonymous referees for their help in improving the quality of this paper, and the Minnesota Supercomputer Institute for providing the computer facilities and an excellent research environment in which to conduct this research.

REFERENCES

[1] L. Adams and J. Ortega, *A multi-color SOR Method for Parallel Computers*, in Proceedings of the 1982 International Conference on Pararallel Processing, 1982, pp. 53–56.

[2] L. M. Adams, *Iterative Algorithms for Large Sparse Linear Systems on Parallel Computers*, Ph.D. thesis, Applied Mathematics, University of Virginia, Charlottesville, VA, 1982. Also NASA Contractor Report 166027, NASA Langley Research Center, Hampton, VA.

[3] E. C. Anderson and Y. Saad, *Solving sparse triangular systems on parallel computers*, Internat. J. High Speed Comput., 1 (1989), pp. 73–96.

[4] O. Axelsson, V. Eijkout, B. Polman, and P. Vassilevski, *Incomplete block-matrix factorization iterative methods for convection-diffusion problems*, BIT, 29 (1989), pp. 867–889.

[5] O. Axelsson and B. Polman, *On approximate factorization methods for block matrices suitable for vector and parallel processors*, Linear Algebra Appl., 77 (1986) pp. 3–26.

[6] M. Benantar and J. E. Flaherty, *A six color procedure for the parallel solution of elliptic systems using the finite quadtree structure*, in Proceedings of the Fourth SIAM Conference on Parallel Processing for Scientific Computing, J. Dongarra, P. Messina, D. C. Sorenson, and R. G. Voigt, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1990, pp. 230–236.

[7] T. F. Coleman and J. J. More, *Estimation of sparse Jacobian matrices and graph coloring problems*, SIAM J. Numer. Anal., 20 (1983), pp. 187–209.

[8] P. Concus, G. H. Golub, and G. Meurant, *Block preconditioning for the conjugate gradient method*, SIAM J. Sci. Stat. Comput., 6 (1985), pp. 309–332.

[9] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, McGraw-Hill, New York, 1990.

[10] P. J. Davis, *Interpolation and Approximation*, Blaisdell, Waltham, MA, 1963.

[11] T. A. Davis, *A Parallel Algorithm for Sparse Unsymmetric LU Factorizations*, Ph.D. thesis, University of Illinois at Urbana Champaign, Urbana, IL, 1989.

[12] E. F. D'Azevedo, F. A. Forsyth, and W. P. Tang, *Ordering methods for preconditioned conjugate gradient methods applied to unstructured grid problems*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 944–961.

[13] ———, *Towards a cost effective ILU preconditioner with high level fill*, BIT, 31 (1992), pp. 442–463.

[14] J. J. Dongarra, I. S. Duff, D. Sorensen, and H. A. van der Vorst, *Solving Linear Systems on Vector and Shared Memory Computers*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1991.

[15] I. S. Duff, A. M. Erisman, and J. K. Reid, *Direct Methods for Sparse Matrices*, Clarendon Press, Oxford, UK, 1986.

[16] I. S. Duff, R. G. Grimes, and J. G. Lewis, *Sparse matrix test problems*, ACM Trans. Math. Software, 15 (1989), pp. 1–14.

[17] ———, *User's Guide for the Harwell-Boeing Sparse Matrix Collection*, Tech. Rep. TR/PA/92/86, CERFACS, Toulouse, France, 1992.

[18] I. S. Duff and G. A. Meurant, *The effect of reordering on preconditioned conjugate gradients*, BIT, 29 (1989), pp. 635–657.

[19] L. C. Dutto, *The effect of reordering on the preconditioned GMRES algorithm for solving the compressible Navier-Stokes equations*, Internat. J. Numer. Methods Engrg., 36 (1993), pp. 457–497.

[20] H. C. Elman and E. Agron, *Ordering techniques for the preconditioned conjugate gradient method on parallel computers*, Comput. Phys. Comm., 53 (1989), pp. 253–269.

[21] H. C. Elman and G. H. Golub, *Iterative methods for cyclically reduced non-self-adjoint linear systems*, Math. Comp., 54 (1990), pp. 671–700.

[22] R. M. Ferencz, *Element-by-Element Preconditioning Techniques for Large Scale Vectorized Finite Element Analysis in Nonlinear Solid and Structural Mechanics*, Ph.D. thesis, Applied Mathematics, Stanford University, CA, 1989.

[23] R. S. Varga and G. H. Golub, *Chebyshev semi iterative methods successive overrelaxation terative methods and second order Richardson iterative methods*, Numer. Math., 3 (1961), pp. 147–168.

[24] K. Gallivan, A. Sameh, and Z. Zlatev, *A parallel hybrid sparse linear system solver*, Comput. Systems Engrg., 1 (1990), pp. 183–195.

[25] J. A. George and J. W. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[26] T. J. R. Hughes, R. M. Ferencz, and J. O. Hallquist, *Large-scale vectorized implicit calculations in solid mechanics on a Cray X-MP/48 utilizing EBE preconditioned conjugate gradients*, Comput. Methods Appl. Mech. Engrg., 61 (1987), pp. 215–248.

[27] O. G. Johnson, C. A. Micchelli, and G. Paul, *Polynomial preconditionings for conjugate gradient calculations*, SIAM J. Numer. Anal., 20 (1983), pp. 362–376.

[28] M. T. Jones and P. E. Plassmann, *Parallel Iterative Solution of Sparse Linear Systems Using Ordering from Graph Coloring Heuristics*, Tech. Rep. MCS-P198-1290, Argonne National Lab., Argonne, IL, 1990.

[29] M. T. JONES AND P. E. PLASSMANN, *A parallel graph coloring heuristic*, Tech. Rep. MCS-P246-0691, Argonne National Lab., Argonne, IL, 1991.

[30] ———, *The effect of many-color orderings on the convergence of methods*, Tech. Rep. MCS-P292-0292, Argonne National Lab., Argonne, IL, 1992.

[31] ———, *Scalable iterative solution of sparse linear systems*, Tech. Rep. MCS-P277-1191, Argonne National Lab., Argonne, IL, 1992.

[32] R. M. KARP, *Reducibility among combinatorial problems*, in Complexity of Computer Computations, Plenum Press, New York, 1972, pp. 85–104.

[33] R. LEUZE, *Independent set orderings for parallel matrix factorizations by Gaussian elimination*, Parallel Comput., 10 (1989), pp. 177–191.

[34] J. G. LEWIS, B. W. PEYTON, AND A. POTHEN, *A fast algorithm for reordering sparse matrices for parallel factorizations*, SIAM J. Sci. Stat. Comput., 6 (1989), pp. 1146–1173.

[35] S. MA AND Y. SAAD, *Distributed ILU*(0) *and SOR Preconditioners for Unstructured Sparse Linear Systems*, Tech. Rep. 94–027, Army High Performance Computing Research Center, University of Minnesota, Minneapolis, MN, 1994.

[36] J. A. MEIJERINK AND H. A. VAN DER VORST, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comp., 31 (1977), pp. 148–162.

[37] J. M. ORTEGA, *Introduction to Parallel and Vector Solution of Linear Systems*, Plenum Press, New York, 1988.

[38] O. OSTERBY AND Z. ZLATEV, *Direct Methods for Sparse Matrices*, Springer-Verlag, New York, 1983.

[39] E. L. POOLE AND J. M. ORTEGA, *Multicolor ICCG methods for vector computers*, SIAM J. Numer. Anal., 24 (1987), pp. 1394–1418.

[40] I. S. DUFF AND J. REID, *The multifrontal solution of unsymmetric sets of linear equations*, SIAM J. Sci. Stat. Comput., 5 (1984), pp. 633–641.

[41] G. RODRIGUE AND D. WOLITZER, *Preconditioning by incomplete block cyclic reduction*, Math. Comp., 42 (1984), pp. 549–565.

[42] Y. SAAD, *Krylov subspace methods on supercomputers*, SIAM J. Sci. Stat. Comput., 10 (1989), pp. 1200–1232.

[43] ———, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM J. Sci. Comput., 14 (1993), pp. 461–469.

[44] ———, *Highly parallel preconditioners for general sparse matrices*, in Recent Advances in Iterative Methods, IMA Volumes in Mathematics and its Applications, Vol. 60, G. Golub, M. Luskin, and A. Greenbaum, eds., Springer-Verlag, Berlin, 1994, pp. 165–199.

[45] ———, *ILUT: A dual threshold incomplete ILU factorization*, Numer. Linear Algebra Appl., 1 (1994), pp. 387–402.

[46] F. SHAKIB, *Finite element analysis of the compressible Euler and Navier Stokes Equations*, Ph.D. thesis, Aeronautics Dept., Stanford University, CA, 1989.

[47] H. A. VAN DER VORST, *A vectorizable version of some ICCG methods*, SIAM J. Sci. Stat. Comput., 3 (1982), pp. 350–356.

[48] H. A. VAN DER VORST AND C. VUIK, *GMRESR: A family of nested GMRES methods*, Numer. Linear Algebra Appl., 1 (1994), pp. 369–386.

[49] R. S. VARGA, *Matrix Iterative Analysis*, Prentice Hall, Englewood Cliffs, NJ, 1962.

[50] V. VENKATAKRISHNAN, H. D. SIMON, AND T. J. BARTH, *A MIMD Implementation of a Parallel Euler Solver for Unstructured Grids*, Tech. Rep. RNR-91-024, NASA Ames Research Center, Moffett Field, CA, 1991.

[51] D. M. YOUNG, *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971.

[52] D. P. YOUNG, R. G. MELVIN, F. T. JOHNSON, J. E. BUSSOLETTI, L. B. WIGTON, AND S. S. SAMANT, *Application of sparse matrix solvers as effective preconditioners*, SIAM J. Sci. Stat. Comput., 10 (1989), pp. 1186–1199.

[53] Z. ZLATEV, *Use of iterative refinement in the solution of sparse linear systems*, SIAM J. Numer. Anal., 19 (1982), pp. 381–399.

SIAM J. SCI. COMPUT.
Vol. 17, No. 4, pp. 848–869, July 1996

EFFICIENT ALGORITHMS FOR COMPUTING A STRONG RANK-REVEALING QR FACTORIZATION*

MING GU† AND STANLEY C. EISENSTAT‡

Abstract. Given an $m \times n$ matrix M with $m \geq n$, it is shown that there exists a permutation Π and an integer k such that the QR factorization

$$M\,\Pi = Q\begin{pmatrix} A_k & B_k \\ & C_k \end{pmatrix}$$

reveals the numerical rank of M: the $k \times k$ upper-triangular matrix A_k is well conditioned, $\|C_k\|_2$ is small, and B_k is linearly dependent on A_k with coefficients bounded by a low-degree polynomial in n. Existing rank-revealing QR (RRQR) algorithms are related to such factorizations and two algorithms are presented for computing them. The new algorithms are nearly as efficient as QR with column pivoting for most problems and take $O(mn^2)$ floating-point operations in the worst case.

Key words. orthogonal factorization, rank-revealing factorization, numerical rank

AMS subject classifications. 65F25, 15A23, 65F35

1. Introduction. Given a matrix $M \in \mathbf{R}^{m\times n}$ with $m \geq n$, we consider partial QR factorizations of the form

$$M\,\Pi = QR \equiv Q\begin{pmatrix} A_k & B_k \\ & C_k \end{pmatrix}, \tag{1}$$

where $Q \in \mathbf{R}^{m\times m}$ is orthogonal, $A_k \in \mathbf{R}^{k\times k}$ is upper triangular with nonnegative diagonal elements, $B_k \in \mathbf{R}^{k\times(n-k)}$, $C_k \in \mathbf{R}^{(m-k)\times(n-k)}$, and $\Pi \in \mathbf{R}^{n\times n}$ is a permutation matrix chosen to reveal linear dependence among the columns of M. Usually k is chosen to be the smallest integer $1 \leq k \leq n$ for which $\|C_k\|_2$ is sufficiently small [24, p. 235].

Golub [20] introduced these factorizations and, with Businger [8], developed the first algorithm (QR with column pivoting) for computing them. Applications include least-squares computations [11, 12, 17, 20, 21, 23, 36], subset selection and linear dependency analysis [12, 18, 22, 34, 44], subspace tracking [7], rank determination [10, 39], and nonsymmetric eigenproblems [2, 15, 26, 35]. Such factorizations are also related to condition estimation [4, 5, 25, 40] and the URV and ULV decompositions [14, 41, 42].

1.1. RRQR factorizations. By the interlacing property of the singular values [24, Cor. 8.3.3], for *any* permutation Π we have[1]

$$\sigma_i(A_k) \leq \sigma_i(M) \quad \text{and} \quad \sigma_j(C_k) \geq \sigma_{k+j}(M) \tag{2}$$

for $1 \leq i \leq k$ and $1 \leq j \leq n-k$. Thus,

$$\sigma_{\min}(A_k) \leq \sigma_k(M) \quad \text{and} \quad \sigma_{\max}(C_k) \geq \sigma_{k+1}(M). \tag{3}$$

Assume that $\sigma_k(M) \gg \sigma_{k+1}(M) \approx 0$, so that the numerical rank of M is k. Then we would like to find a Π for which $\sigma_{\min}(A_k)$ is sufficiently large and $\sigma_{\max}(C_k)$ is sufficiently

*Received by the editors May 13, 1994; accepted for publication (in revised form) March 8, 1995. This research was supported in part by U. S. Army Research Office contract DAAL03-91-G-0032.

†Department of Mathematics and Lawrence Berkeley Laboratory, University of California, Berkeley, CA 94720 (minggu@math.berkeley.edu).

‡Department of Computer Science, Yale University, P. O. Box 208285, New Haven, CT 06520-8285 (eisenstat-stan@cs.yale.edu).

[1] Here $\sigma_i(X)$, $\sigma_{\max}(X)$, and $\sigma_{\min}(X)$ denote the ith largest, the largest, and the smallest singular values of the matrix X, respectively.

small. We call the factorization (1) a rank-revealing QR (RRQR) factorization if it satisfies (cf. (3))

$$\sigma_{\min}(A_k) \geq \frac{\sigma_k(M)}{p(k,n)} \quad \text{and} \quad \sigma_{\max}(C_k) \leq \sigma_{k+1}(M)\; p(k,n), \tag{4}$$

where $p(k,n)$ is a function bounded by a low-degree polynomial in k and n [13, 28]. Other, less restrictive definitions are discussed in [13] and [37]. The term "rank-revealing QR factorization" is due to Chan [10].

The Businger and Golub algorithm [8, 20] works well in practice, but there are examples where it fails to produce a factorization satisfying (4) (see Example 1 in §2). Other algorithms fail on similar examples [13]. Recently, Hong and Pan [28] showed that there exist RRQR factorizations with $p(k,n) = \sqrt{k(n-k) + \min(k, n-k)}$, and Chandrasekaran and Ipsen [13] developed an algorithm that computes one efficiently in practice,[2] given k.

1.2. Strong RRQR factorizations. In some applications it is necessary to find a basis for the approximate right null space of M, as in rank-deficient least-squares computations [23, 24] and subspace tracking [7], or to separate the linearly independent columns of M from the linearly dependent ones, as in subset selection and linear dependency analysis [12, 18, 22, 34, 44]. The RRQR factorization does not lead to a stable algorithm because the elements of $A_k^{-1}B_k$ can be very large (see Example 2 in §2).

In this paper we show that there exist QR factorizations that meet this need. We call the factorization (1) a *strong* RRQR factorization if it satisfies (cf. (2))

$$\sigma_i(A_k) \geq \frac{\sigma_i(M)}{q_1(k,n)} \quad \text{and} \quad \sigma_j(C_k) \leq \sigma_{k+j}(M)\; q_1(k,n) \tag{5}$$

and

$$\left|\left(A_k^{-1}B_k\right)_{i,j}\right| \leq q_2(k,n), \tag{6}$$

for $1 \leq i \leq k$ and $1 \leq j \leq n-k$, where $q_1(k,n)$ and $q_2(k,n)$ are functions bounded by low-degree polynomials in k and n. Clearly a strong RRQR factorization is also a RRQR factorization. In addition, condition (6) makes

$$\Pi \begin{pmatrix} -A_k^{-1}B_k \\ I_{n-k} \end{pmatrix}$$

an approximate right null space of M with a small residual independent of the condition number of A_k, provided that A_k is not too ill conditioned [38, pp. 192–198]. See [26] for another application.

We show that there exists a permutation Π for which conditions (5) and (6) hold with

$$q_1(k,n) = \sqrt{1 + k(n-k)} \quad \text{and} \quad q_2(k,n) = 1.$$

Since this permutation might take exponential time to compute, we present algorithms that, given $f \geq 1$, find a Π for which (5) and (6) hold with

$$q_1(k,n) = \sqrt{1 + f^2 k(n-k)} \quad \text{and} \quad q_2(k,n) = f.$$

Here k can be either an input parameter (Algorithm 4) or the smallest integer for which $\sigma_{\max}(C_k)$ is sufficiently small (Algorithm 5). When $f > 1$, these algorithms require $O\left((m + n\log_f n)n^2\right)$ floating-point operations. In particular, when f is a small power of n (e.g., $\sqrt{n}$ or n), they take $O(mn^2)$ time (see §4.4).

[2] In the worst case the runtime might be exponential in k or n. The algorithm proposed by Golub, Klema, and Stewart [22] also computes an RRQR factorization [30], but requires an orthogonal basis for the right null space.

Recently, Pan and Tang [37] presented an algorithm that, given $f > 1$, computes an RRQR factorization with $p(k,n) = f\sqrt{k(n-k)} + \max(k, n-k)$. This algorithm can be shown to be mathematically equivalent to Algorithm 5 and thus computes a strong RRQR factorization with $q_1(k,n) = \sqrt{1 + f^2k(n-k)}$ and $q_2(k,n) = f$. However, it is much less efficient. Pan and Tang [37] also present two practical modifications to their algorithm, but they do not always compute strong RRQR factorizations.

1.3. Overview. In §2 we review QR with column pivoting [8, 20] and the Chandrasekaran and Ipsen algorithm [13] for computing an RRQR factorization. In §3 we give a constructive existence proof for the strong RRQR factorization. In §4 we present an algorithm (Algorithm 5) that computes a strong RRQR factorization and bound the total number of operations required when $f > 1$; and in §5 we show that this algorithm is numerically stable. In §6 we report the results of some numerical experiments. In §7 we show that the concept of a strong RRQR factorization is not completely new in that the QR factorization given by the Businger and Golub algorithm [8, 20] satisfies (5) and (6) with $q_1(k,n)$ and $q_2(k,n)$ functions that grow exponentially with k. Finally, in §8 we present some extensions of this work, including a version of Algorithm 5 that is nearly as fast as QR with column pivoting for most problems and takes $O(mn^2)$ floating-point operations in the worst case.

1.4. Notation. By convention, A_k, $\bar{A}_k \in \mathbf{R}^{k\times k}$ denote upper-triangular matrices with nonnegative diagonal elements, and B_k, $\bar{B}_k \in \mathbf{R}^{k\times(n-k)}$ and C_k, $\bar{C}_k \in \mathbf{R}^{(m-k)\times(n-k)}$ denote general matrices.

In the partial QR factorization

$$X = Q\begin{pmatrix} A_k & B_k \\ & C_k \end{pmatrix}$$

of a matrix $X \in \mathbf{R}^{m\times n}$ (where the diagonal elements of A_k are nonnegative), we write

$$\mathcal{A}_k(X) = A_k, \quad \mathcal{C}_k(X) = C_k, \quad \text{and} \quad \mathcal{R}_k(X) = \begin{pmatrix} A_k & B_k \\ & C_k \end{pmatrix}.$$

For A, a nonsingular $\ell \times \ell$ matrix, $1/\omega_i(A)$ denotes the 2-norm of the ith *row* of A^{-1} and $\omega_*(A) = (\omega_1(A), \ldots, \omega_\ell(A))^T$. For C, a matrix with ℓ columns, $\gamma_j(C)$ denotes the 2-norm of the jth *column* of C and $\gamma_*(C) = (\gamma_1(C), \ldots, \gamma_\ell(C))$.

$\Pi_{i,j}$ denotes the permutation that interchanges the ith and jth columns of a matrix.

A *flop* is a floating-point operation $\alpha \circ \beta$, where α and β are floating-point numbers and $\circ$ is one of $+$, $-$, $\times$, and $\div$. Taking the absolute value or comparing two floating-point numbers is also counted as a flop.

2. RRQR algorithms. QR with column pivoting [8, 20] is a modification of the ordinary QR algorithm.

ALGORITHM 1. QR with column pivoting.

```
$k := 0$; $R := M$; $\Pi := I$;
while $\max_{1\le j\le n-k} \gamma_j\,(\mathcal{C}_k(R)) \ge \delta$ do
    $j_{\max} := \operatorname{argmax}_{1\le j\le n-k}\ \gamma_j\,(\mathcal{C}_k(R))$;
    $k := k+1$;
    Compute $R := \mathcal{R}_k(R\,\Pi_{k,k+j_{\max}-1})$ and $\Pi := \Pi\,\Pi_{k,k+j_{\max}-1}$;
endfor;
```

When Algorithm 1 halts, we have

$$\sigma_{\max}\,(\mathcal{C}_k(M\,\Pi)) \le \sqrt{n-k}\ \max_{1\le j\le n-k} \gamma_j\,(\mathcal{C}_k(M\,\Pi)) \le \sqrt{n-k}\ \delta,$$

and if δ is sufficiently small, then the numerical rank of M is at most k. If the vector of column norms $\gamma_*\ (\mathcal{C}_k(R))$ is updated rather than recomputed from scratch each time, then Algorithm 1 takes about $4mnk - 2k^2(m+n) + 4k^3/3$ flops [24, p. 236].

Algorithm 1 uses a greedy strategy for finding well-conditioned columns: having determined the first k columns, it picks a column from the remaining $n-k$ columns that maximizes $\det\left[\mathcal{A}_{k+1}(R)\right]$ (see [13]). When there are only a few well-conditioned columns, this strategy is guaranteed to find a strong RRQR factorization (see §7). It also works well in general, but it fails to find an RRQR factorization for the following example.

Example 1 (Kahan [33]). Let $M = S_n K_n$, where

$$(7) \qquad S_n = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & \varsigma & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \varsigma^{n-1} \end{pmatrix} \quad \text{and} \quad K_n = \begin{pmatrix} 1 & -\varphi & \cdots & -\varphi \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & -\varphi \\ 0 & \cdots & 0 & 1 \end{pmatrix},$$

with $\varphi, \varsigma > 0$ and $\varphi^2 + \varsigma^2 = 1$. Let $k = n-1$. Then Algorithm 1 does not permute the columns of M, yet it can be shown that

$$\frac{\sigma_k(M)}{\sigma_{\min}(A_k)} \geq \frac{\varphi^3(1+\varphi)^{n-4}}{2\varsigma},$$

and the right-hand side grows faster than any polynomial in k and n.

When $m = n$ and the numerical rank of M is close to n, Stewart [39] suggests applying Algorithm 1 to M^{-1}. Recently, Chandrasekaran and Ipsen [13] combined these ideas to construct an algorithm Hybrid-III(k) that is guaranteed to find an RRQR factorization, given k. We present it in a different form here to motivate our constructive proof of the existence of a strong RRQR factorization.

```
ALGORITHM 2. Hybrid-III(k).
  $R := M$; $\Pi := I$;
  repeat
    $i_{\min} := \mathrm{argmin}_{1 \leq i \leq k}\ \omega_i\ (\mathcal{A}_k(R))$;
    if there exists a $j$ such that $\det\left[\mathcal{A}_k(R\,\Pi_{i_{\min},j+k})\right] / \det\left[\mathcal{A}_k(R)\right] > 1$ then
      Find such a $j$;
      Compute $R := \mathcal{R}_k(R\,\Pi_{i_{\min},j+k})$ and $\Pi := \Pi\,\Pi_{i_{\min},j+k}$;
    endif;
    $j_{\max} := \mathrm{argmax}_{1 \leq j \leq n-k}\ \gamma_j\ (\mathcal{C}_k(R))$;
    if there exists an $i$ such that $\det\left[\mathcal{A}_k(R\,\Pi_{i,j_{\max}+k})\right] / \det\left[\mathcal{A}_k(R)\right] > 1$ then
      Find such an $i$;
      Compute $R := \mathcal{R}_k(R\,\Pi_{i,j_{\max}+k})$ and $\Pi := \Pi\,\Pi_{i,j_{\max}+k}$;
    endif;
  until no interchange occurs;
```

Since the objective is to find a permutation Π for which $\sigma_{\min}\ (\mathcal{A}_k(M\,\Pi))$ is sufficiently large and $\sigma_{\max}\ (\mathcal{C}_k(M\,\Pi))$ is sufficiently small, Algorithm 2 keeps interchanging the most "dependent" of the first k columns (column $i_{\min}$) with one of the last $n-k$ columns, and interchanging the most "independent" of the last $n-k$ columns (column $j_{\max}$) with one of the first k columns, as long as $\det\left[\mathcal{A}_k(R)\right]$ strictly increases.

Since $\det[\mathcal{A}_k(R)]$ strictly increases with every interchange, no permutation repeats; and since there are only a finite number of permutations, Algorithm 2 eventually halts. Chandrasekaran and Ipsen [13] also show that it computes an RRQR factorization, given k. Due to efficiency considerations, they suggest that it be run as a postprocessor to Algorithm 1.

But Algorithm 2 may not compute a strong RRQR factorization either.

Example 2. Let $k = n - 2$ and let

$$M \equiv \begin{pmatrix} A_k & B_k \\ & C_k \end{pmatrix} = \begin{pmatrix} S_{k-1}K_{k-1} & 0 & 0 & -\varphi S_{k-1}c_{k-1} \\ & \mu & 0 & 0 \\ & & \mu & 0 \\ & & & \mu \end{pmatrix},$$

where S_{k-1} and K_{k-1} are defined as in (7), $c_{k-1} = (1, \ldots, 1)^T \in \mathbf{R}^{k-1}$, and

$$\mu = \frac{1}{\sqrt{k}} \min_{1 \le i \le k-1} \omega_i(S_{k-1}K_{k-1}).$$

Then Algorithm 2 does not permute the columns of M (note that $i_{\min} = k$ and $j_{\max} = k + 1$), yet it can be shown that

$$\frac{\sigma_{k-1}(M)}{\sigma_{k-1}(A_k)} \ge \frac{\varphi^3(1+\varphi)^{k-4}}{2\varsigma} \quad \text{and} \quad \|A_k^{-1}B_k\|_\infty = \varphi(1+\varphi)^{k-2},$$

and the right-hand sides grow faster than any polynomial in k and n.

Since Algorithm 1 does not permute the columns of M, this example also shows that Algorithm 2 may not compute a strong RRQR factorization even when it is run as a postprocessor to Algorithm 1.

3. The existence of a strong RRQR factorization.

A strong RRQR factorization satisfies three conditions: *every* singular value of A_k is sufficiently large, *every* singular value of C_k is sufficiently small, and *every* element of $A_k^{-1}B_k$ is bounded. Since

$$\det(A_k) = \prod_{i=1}^{k} \sigma_i(A_k) = \sqrt{\det(M^TM)} \Bigg/ \prod_{j=1}^{n-k} \sigma_j(C_k),$$

a strong RRQR factorization also results in a large $\det(A_k)$. Given k and $f \ge 1$, Algorithm 3 below[3] constructs a strong RRQR factorization by using column interchanges to try to maximize $\det(A_k)$.

```
ALGORITHM 3. Compute a strong RRQR factorization, given $k$.
  $R := \mathcal{R}_k(M)$; $\Pi := I$;
  while there exist $i$ and $j$ such that $\det(\bar{A}_k))/\det(A_k) > f$,
     where $R = \begin{pmatrix} A_k & B_k \\ & C_k \end{pmatrix}$ and $\mathcal{R}_k(R\,\Pi_{i,j+k}) = \begin{pmatrix} \bar{A}_k & \bar{B}_k \\ & \bar{C}_k \end{pmatrix}$, do
     Find such an $i$ and $j$;
     Compute $R := \mathcal{R}_k(R\,\Pi_{i,j+k})$ and $\Pi := \Pi\,\Pi_{i,j+k}$;
  endwhile;
```

While Algorithm 2 interchanges either the most "dependent" column of A_k or the most "independent" column of C_k, Algorithm 3 interchanges any pair of columns that sufficiently increases $\det(A_k)$. As before, there are only a finite number of permutations and none can repeat, so that it eventually halts.

[3] The algorithms in this section are only intended to prove the existence of a strong RRQR factorization. Efficient algorithms will be presented in §§4 and 8.

To prove that Algorithm 3 computes a strong RRQR factorization, we first express $\det(\bar{A}_k)/\det(A_k)$ in terms of $\omega_i(A_k)$, $\gamma_j(C_k)$, and $\left(A_k^{-1}B_k\right)_{i,j}$.

LEMMA 3.1. *Let*

$$R = \begin{pmatrix} A_k & B_k \\ & C_k \end{pmatrix} \quad \textit{and} \quad \mathcal{R}_k(R\,\Pi_{i,j+k}) = \begin{pmatrix} \bar{A}_k & \bar{B}_k \\ & \bar{C}_k \end{pmatrix},$$

where A_k *has positive diagonal elements. Then*

$$\frac{\det(\bar{A}_k)}{\det(A_k)} = \sqrt{\left(A_k^{-1}B_k\right)_{i,j}^2 + \left(\gamma_j(C_k)/\omega_i(A_k)\right)^2}.$$

Proof. First, assume that $i < k$ or that $j > 1$. Let $A_k\,\Pi_{i,k} = \tilde{Q}\tilde{A}_k$ be the QR factorization of $A_k\,\Pi_{i,k}$, let $\tilde{B}_k = \tilde{Q}^T B_k\,\Pi_{1,j}$ and $\tilde{C}_k = C_k\,\Pi_{1,j}$, and let $\tilde{\Pi} = \operatorname{diag}(\Pi_{i,k}, \Pi_{1,j})$. Then

$$R\,\tilde{\Pi} \equiv \begin{pmatrix} A_k\,\Pi_{i,k} & B_k\,\Pi_{1,j} \\ & C_k\,\Pi_{1,j} \end{pmatrix} = \begin{pmatrix} \tilde{Q} & \\ & I_{m-k} \end{pmatrix} \begin{pmatrix} \tilde{A}_k & \tilde{B}_k \\ & \tilde{C}_k \end{pmatrix}$$

is the QR factorization of $R\,\tilde{\Pi}$. Since both A_k and $\tilde{A}_k$ have positive diagonal elements, we have $\det(A_k) = \det(\tilde{A}_k)$. Since $\tilde{A}_k^{-1}\tilde{B}_k = \Pi_{i,k}^T A_k^{-1} B_k \Pi_{1,j}$, we have $(A_k^{-1}B_k)_{i,j} = (\tilde{A}_k^{-1}\tilde{B}_k)_{k,1}$. Since $\tilde{A}_k^{-1} = \Pi_{i,k}^T A_k^{-1} B_k \tilde{Q}$ and postmultiplication by an orthogonal matrix leaves the 2-norms of the rows unchanged, we have $\omega_i(A_k) = \omega_k(\tilde{A}_k)$. Finally, we have $\gamma_j(C_k) = \gamma_1(\tilde{C}_k)$. Thus it suffices to consider the special case $i = k$ and $j = 1$.

Partition

$$\mathcal{R}_{k+1}(R) = \begin{pmatrix} A_{k-1} & b_1 & b_2 & B \\ & \gamma_1 & \beta & c_1^T \\ & & \gamma_2 & c_2^T \\ & & & C_{k+1} \end{pmatrix}.$$

Then $\omega_i(A_k) = \gamma_1$, $\gamma_j(C_k) = \gamma_2$, and $\left(A_k^{-1}B_k\right)_{i,j} = \beta/\gamma_1$. But $\det(A_k) = \det(A_{k-1})\,\gamma_1$ and $\det(\bar{A}_k) = \det(A_{k-1})\sqrt{\beta^2 + \gamma_2^2}$, so that

$$\frac{\det(\bar{A}_k)}{\det(A_k)} = \sqrt{(\beta/\gamma_1)^2 + (\gamma_2/\gamma_1)^2} = \sqrt{\left(A_k^{-1}B_k\right)_{i,j}^2 + \left(\gamma_j(C_k)/\omega_i(A_k)\right)^2},$$

which is the result required. □

Let

$$\rho(R,k) = \max_{1\le i\le k,\ 1\le j\le n-k} \sqrt{\left(A_k^{-1}B_k\right)_{i,j}^2 + \left(\gamma_j(C_k)/\omega_i(A_k)\right)^2}.$$

Then by Lemma 3.1, Algorithm 3 can be rewritten as the following.

ALGORITHM 4. Compute a strong RRQR factorization, given k.

```
Compute $R \equiv \begin{pmatrix} A_k & B_k \\ & C_k \end{pmatrix} := \mathcal{R}_k(M)$ and $\Pi = I$;
while $\rho(R,k) > f$ do
    Find $i$ and $j$ such that $\sqrt{\left(A_k^{-1}B_k\right)_{i,j}^2 + \left(\gamma_j(C_k)/\omega_i(A_k)\right)^2} > f$;
    Compute $R \equiv \begin{pmatrix} A_k & B_k \\ & C_k \end{pmatrix} := \mathcal{R}_k(R\,\Pi_{i,j+k})$ and $\Pi := \Pi\,\Pi_{i,j+k}$;
endwhile;
```

Since Algorithm 4 is equivalent to Algorithm 3, it eventually halts and finds a permutation Π for which $\rho(\mathcal{R}_k(M\,\Pi), k) \le f$. This implies (6) with $q_2(k,n) = f$. Now we show that this also implies (5) with $q_1(k,n) = \sqrt{1 + f^2 k(n-k)}$, i.e., that Algorithms 3 and 4 compute a strong RRQR factorization, given k.

THEOREM 3.2. *Let*

$$R \equiv \begin{pmatrix} A_k & B_k \\ & C_k \end{pmatrix} = \mathcal{R}_k(M\,\Pi)$$

satisfy $\rho(R,k) \le f$. *Then*

$$\sigma_i(A_k) \ge \frac{\sigma_i(M)}{\sqrt{1 + f^2 k(n-k)}}, \qquad 1 \le i \le k, \tag{8}$$

and

$$\sigma_j(C_k) \le \sigma_{j+k}(M)\,\sqrt{1 + f^2 k(n-k)}, \qquad 1 \le j \le n-k. \tag{9}$$

Proof. For simplicity we assume that M (and therefore R) has full column rank. Let $\alpha = \sigma_{\max}(C_k)/\sigma_{\min}(A_k)$, and write

$$R = \begin{pmatrix} A_k & \\ & C_k/\alpha \end{pmatrix} \begin{pmatrix} I_k & A_k^{-1}B_k \\ & \alpha I_{n-k} \end{pmatrix} \equiv \tilde{R}_1 W_1.$$

Then by [29, Thm. 3.3.16],

$$\sigma_i(R) \le \sigma_i(\tilde{R}_1)\; \|W_1\|_2, \qquad 1 \le i \le n. \tag{10}$$

Since $\sigma_{\min}(A_k) = \sigma_{\max}(C_k/\alpha)$, we have $\sigma_i(\tilde{R}_1) = \sigma_i(A_k)$ for $1 \le i \le k$. Moreover,

$$\begin{aligned}
\|W_1\|_2^2 &\le 1 + \|A_k^{-1}B_k\|_2^2 + \alpha^2 \\
&= 1 + \|A_k^{-1}B_k\|_2^2 + \|C_k\|_2^2 \|A_k^{-1}\|_2^2 \\
&\le 1 + \|A_k^{-1}B_k\|_F^2 + \|C_k\|_F^2 \|A_k^{-1}\|_F^2 \\
&= 1 + \sum_{i=1}^{k} \sum_{j=1}^{n-k} \left\{ \left(A_k^{-1}B_k\right)_{i,j}^2 + \gamma_j(C_k)^2/\omega_i(A_k)^2 \right\} \\
&\le 1 + f^2 k(n-k),
\end{aligned}$$

so that $\|W_1\|_2 \le \sqrt{1 + f^2 k(n-k)}$. Plugging these relations into (10), we get (8). Similarly, let

$$\tilde{R}_2 \equiv \begin{pmatrix} \alpha A_k & \\ & C_k \end{pmatrix} = \begin{pmatrix} A_k & B_k \\ & C_k \end{pmatrix} \begin{pmatrix} \alpha I_k & -A_k^{-1}B_k \\ & I_{n-k} \end{pmatrix} \equiv R W_2.$$

Then

$$\sigma_j(C_k) = \sigma_{j+k}(\tilde{R}_2) \le \sigma_{j+k}(R) \|W_2\|_2 \le \sigma_{j+k}(M)\,\sqrt{1 + f^2 k(n-k)},$$

which is (9). □

4. Computing a strong RRQR factorization. Given $f \ge 1$ and a tolerance $\delta > 0$, Algorithm 5 below computes both k and a strong RRQR factorization. It is a combination of the ideas in Algorithms 1 and 4 but uses

$$\hat{\rho}(R,k) = \max_{1 \le i \le k,\ 1 \le j \le n-k} \max\left\{ \left| \left(A_k^{-1}B_k\right)_{i,j} \right|,\ \gamma_j(C_k)/\omega_i(A_k) \right\}$$

instead of $\rho(R,k)$ and computes $\omega_*(A_k)$, $\gamma_*(C_k)$, and $A_k^{-1}B_k$ recursively for greater efficiency.

ALGORITHM 5. Compute k and a strong RRQR factorization.

```
$k := 0$; $R \equiv C_k := M$; $\Pi := I$;
Initialize $\omega_*(A_k)$, $\gamma_*(C_k)$, and $A_k^{-1}B_k$;
while $\max_{1\le j\le n-k} \gamma_j(C_k) \ge \delta$ do
    $j_{\max} := \operatorname{argmax}_{1\le j\le n-k}\ \gamma_j(C_k)$;
    $k := k+1$;
    Compute $R \equiv \begin{pmatrix} A_k & B_k \\ & C_k \end{pmatrix} := \mathcal{R}_k(R\,\Pi_{k,k+j_{\max}-1})$ and $\Pi := \Pi\,\Pi_{k,k+j_{\max}-1}$;
    Update $\omega_*(A_k)$, $\gamma_*(C_k)$, and $A_k^{-1}B_k$;
    while $\hat{\rho}(R,k) > f$ do
        Find $i$ and $j$ such that $\left|\left(A_k^{-1}B_k\right)_{i,j}\right| > f$ or $\gamma_j(C_k)/\omega_i(A_k) > f$;
        Compute $R \equiv \begin{pmatrix} A_k & B_k \\ & C_k \end{pmatrix} := \mathcal{R}_k(R\,\Pi_{i,j+k})$ and $\Pi := \Pi\,\Pi_{i,j+k}$;
        Modify $\omega_*(A_k)$, $\gamma_*(C_k)$, and $A_k^{-1}B_k$;
    endwhile;
endwhile;
```

Since its inner **while**-loop is essentially equivalent to Algorithm 4, Algorithm 5 must eventually halt, having found k and a permutation Π for which $\hat{\rho}(R,k) \le f$. This implies that $\rho(\mathcal{R}_k(M\,\Pi),k) \le \sqrt{2}\,f$, so that (5) and (6) are satisfied with[4] $q_1(k,n) = \sqrt{1+2f^2k(n-k)}$ and $q_2(k,n) = \sqrt{2}f$.

Remark 1. Note that

$$\frac{\sigma_{k+1}(M)}{\sigma_k(M)} \ge \frac{1}{q_1(k,n)^2}\,\frac{\sigma_{\max}(C_k)}{\sigma_{\min}(A_k)} \ge \frac{1}{q_1(k,n)^2} \max_{1\le i\le k,\ 1\le j\le n-k} \frac{\gamma_j(C_k)}{\omega_i(A_k)}$$

and

$$\frac{\sigma_{k+1}(M)}{\sigma_k(M)} \le \frac{\sigma_{\max}(C_k)}{\sigma_{\min}(A_k)} \le \sqrt{k(n-k)} \max_{1\le i\le k,\ 1\le j\le n-k} \frac{\gamma_j(C_k)}{\omega_i(A_k)}.$$

Thus Algorithm 5 can detect a sufficiently large gap in the singular values of M if we change the condition in the outer **while**-loop to

$$\max_{1\le j\le n-k} \gamma_j(C_k) \ge \delta \quad \textbf{or} \quad \max_{1\le i\le k,\ 1\le j\le n-k} \gamma_j(C_k)/\omega_i(A_k) \ge \zeta,$$

where ζ is some tolerance. This is useful when solving rank-deficient least-squares problems using RRQR factorizations (see [11, 12] and the references therein).

In §§4.1–4.3 we show how to update A_k, B_k, C_k, $\omega_*(A_k)$, $\gamma_*(C_k)$, and $A_k^{-1}B_k$ after k increases and to modify them after an interchange. In §4.4 we bound the total number of interchanges and the total number of operations. We will discuss numerical stability in §5.

4.1. Updating formulas.

Let

$$R = \begin{pmatrix} A_{k-1} & B_{k-1} \\ & C_{k-1} \end{pmatrix} \quad \text{and} \quad \mathcal{R}_k(R\,\Pi_{k,k+j_{\max}-1}) = \begin{pmatrix} A_k & B_k \\ & C_k \end{pmatrix}.$$

Assume that we have already computed A_{k-1}, B_{k-1}, C_{k-1}, $\omega_*(A_{k-1})$, $\gamma_*(C_{k-1})$, and $A_{k-1}^{-1}B_{k-1}$. In this subsection we show how to compute A_k, B_k, C_k, $\omega_*(A_k)$, $\gamma_*(C_k)$, and $A_k^{-1}B_k$. For simplicity we assume that $j_{\max} = 1$, so that $\gamma_1(C_{k-1}) \ge \gamma_j(C_{k-1})$ for $1 \le j \le n-k+1$.

[4] To get $q_1(k,n) = \sqrt{1+f^2k(n-k)}$ and $q_2(k,n) = f$, replace $\hat{\rho}(R,k)$ by $\rho(R,k)$ or replace f by $f/\sqrt{2}$ (assuming that $f > \sqrt{2}$) in Algorithm 5.

Let $H \in \mathbf{R}^{(m-k)\times(m-k)}$ be an orthogonal matrix that zeroes out the elements below the diagonal in the first column of C_{k-1}, and let

$$B_{k-1} = (\, b \quad B\,) \quad \text{and} \quad HC_{k-1} = \begin{pmatrix} \gamma & c^T \\ & C \end{pmatrix},$$

where $\gamma = \gamma_1(C_{k-1})$. Then

$$\begin{pmatrix} A_k & B_k \\ & C_k \end{pmatrix} = \begin{pmatrix} A_{k-1} & b & B \\ & \gamma & c^T \\ & & C \end{pmatrix},$$

so that

$$A_k = \begin{pmatrix} A_{k-1} & b \\ & \gamma \end{pmatrix}, \quad B_k = \begin{pmatrix} B \\ c^T \end{pmatrix}, \quad \text{and} \quad C_k = C.$$

Let $A_{k-1}^{-1}B_{k-1} = (\, u \quad U\,)$. Then

$$A_k^{-1} = \begin{pmatrix} A_{k-1}^{-1} & -u/\gamma \\ & 1/\gamma \end{pmatrix} \quad \text{and} \quad A_k^{-1}B_k = \begin{pmatrix} U - uc^T/\gamma \\ c^T/\gamma \end{pmatrix}.$$

Let $u = (\mu_1, \ldots, \mu_{k-1})^T$ and $c = (\nu_1, \ldots, \nu_{n-k})^T$. Then $\omega_*(A_k)$ and $\gamma_*(C_k)$ can be computed from

$$\omega_k(A_k) = \gamma \quad \text{and} \quad 1/\omega_i(A_k)^2 = 1/\omega_i(A_{k-1})^2 + \mu_i^2/\gamma^2, \quad 1 \le i \le k-1$$

and

$$\gamma_j(C_k)^2 = \gamma_{j+1}(C_{k-1})^2 - \nu_j^2, \quad 1 \le j \le n-k.$$

The main cost of the updating procedure is in computing HC_{k-1} and $U - uc^T/\gamma$, which take about $4(m-k)(n-k)$ and $2k(n-k)$ flops, respectively, for a total of about $2(2m-k)(n-k)$ flops.

Remark 2. Since $f \ge 1$, $\rho(R, k-1) \le \sqrt{2}\, f$, and $\gamma \ge \gamma_{j+1}(C_{k-1}) \ge \nu_j$, for $1 \le j \le n-k$, we have

$$\left|\left(A_k^{-1}B_k\right)_{i,j}\right| \le 2f \quad \text{and} \quad \gamma_j(C_k)/\omega_i(A_k) \le \sqrt{2}\, f,$$

so that

$$\rho(\mathcal{R}_k(R\,\Pi_{k,j_{\max}}), k) \le \sqrt{6}\, f.$$

This bound will be used in §5.1.

4.2. Reducing a general interchange to a special one. Assume that there is an interchange between the ith and $(j+k)$th columns of R. In this subsection we show how to reduce this to the special case $i = k$ and $j = 1$.

Let

$$R = \begin{pmatrix} A_k & B_k \\ & C_k \end{pmatrix}.$$

If $j > 1$, then interchange the $(k+1)$st and $(k+j)$th columns of R. This only interchanges the corresponding columns in B_k, C_k, $\gamma_*(C_k)$, and $A_k^{-1}B_k$. Henceforth we assume that $i < k$ and $j = 1$.

Partition

$$A_k = \begin{pmatrix} A_{1,1} & a_1 & A_{1,2} \\ & \alpha & a_2^T \\ & & A_{2,2} \end{pmatrix},$$

where $A_{1,1} \in \mathbf{R}^{(i-1)\times(i-1)}$ and $A_{2,2} \in \mathbf{R}^{(k-i)\times(k-i)}$ are upper triangular. Let Π_k be the permutation that cyclically shifts the last $k - i + 1$ columns of A_k to the left, so that

$$A_k\,\Pi_k = \begin{pmatrix} A_{1,1} & A_{1,2} & a_1 \\ & a_2^T & \alpha \\ & A_{2,2} & \end{pmatrix}.$$

Note that $A_k\,\Pi_k$ is an upper-Hessenberg matrix with nonzero subdiagonal elements in columns $i, i+1, \ldots, k-1$.

To retriangularize $A_k\,\Pi_k$, we apply Givens rotations to successively zero out the nonzero subdiagonal elements in columns $i, i+1, \ldots, k-1$ (see [19, 24]). Let Q_k^T be the product of these Givens rotations, so that $Q_k^T A_k\,\Pi_k$ is upper triangular.

Let $\tilde{\Pi} = \operatorname{diag}(\Pi_k, I_{n-k})$, so that the ith column of R is the kth column of $R\,\tilde{\Pi}$. Then

$$R\,\tilde{\Pi} = \begin{pmatrix} A_k\,\Pi_k & B_k \\ & C_k \end{pmatrix} \quad \text{and} \quad \mathcal{R}_k(R\,\tilde{\Pi}) \equiv \begin{pmatrix} \bar{A}_k & \bar{B}_k \\ & \bar{C}_k \end{pmatrix} = \begin{pmatrix} Q_k^T A_k\,\Pi_k & Q_k^T B_k \\ & C_k \end{pmatrix}.$$

Since $\bar{A}_k^{-1} = \Pi_k^T A_k^{-1} Q_k$ and postmultiplication by an orthogonal matrix leaves the 2-norms of the rows unchanged, it follows that

$$\omega_*(\bar{A}_k) = \Pi_k^T\,\omega_*(A_k), \quad \gamma_*(\bar{C}_k) = \gamma_*(C_k), \quad \text{and} \quad \bar{A}_k^{-1}\bar{B}_k = \Pi_k^T\left(A_k^{-1}B_k\right).$$

The main cost of this reduction is in computing $Q_k^T A_k\,\Pi_k$ and $Q_k^T B_k$, which takes about $3\left((n-i)^2 - (n-k)^2\right) \le 3k(2n-k)$ flops.

4.3. Modifying formulas. In this subsection we show how to modify A_k, B_k, C_k, $\omega_*(A_k)$, $\gamma_*(C_k)$, and $A_k^{-1}B_k$ when there is an interchange between the kth and $(k+1)$st columns of R. We assume that we have already zeroed out the elements below the diagonal in the $(k+1)$st column.

Writing

$$R \equiv \begin{pmatrix} A_k & B_k \\ & C_k \end{pmatrix} = \begin{pmatrix} A_{k-1} & b_1 & b_2 & B \\ & \gamma & \gamma\mu & c_1^T \\ & & \gamma\nu & c_2^T \\ & & & C_{k+1} \end{pmatrix},$$

we have

$$\mathcal{R}_{k+1}(R\,\Pi_{k,k+1}) \equiv \begin{pmatrix} \bar{A}_k & \bar{B}_k \\ & \bar{C}_k \end{pmatrix} = \begin{pmatrix} A_{k-1} & b_2 & b_1 & B \\ & \bar{\gamma} & \gamma\mu/\rho & \bar{c}_1^T \\ & & \gamma\nu/\rho & \bar{c}_2^T \\ & & & C_{k+1} \end{pmatrix},$$

where $\rho = \sqrt{\mu^2 + \nu^2}$, $\bar{\gamma} = \gamma\rho$, $\bar{c}_1 = (\mu c_1 + \nu c_2)/\rho$, and $\bar{c}_2 = (\nu c_1 - \mu c_2)/\rho$.

From the expression for R, we also have

$$A_k^{-1} = \begin{pmatrix} A_{k-1}^{-1} & -u/\gamma \\ & 1/\gamma \end{pmatrix},$$

where $u = A_{k-1}^{-1} b_1$. Since A_{k-1} is upper triangular, we can compute u using back-substitution. Moreover,

$$A_k^{-1} B_k \equiv \begin{pmatrix} u_1 & U \\ \mu & u_2^T \end{pmatrix} = \begin{pmatrix} A_{k-1}^{-1} & -u/\gamma \\ & 1/\gamma \end{pmatrix} \begin{pmatrix} b_2 & B \\ \gamma\mu & c_1^T \end{pmatrix},$$

so that

$$A_{k-1}^{-1} b_2 = u_1 + \mu u \quad \text{and} \quad A_{k-1}^{-1} B = U + u c_1^T/\gamma.$$

It follows that

$$\bar{A}_k^{-1} = \begin{pmatrix} A_{k-1}^{-1} & -A_{k-1}^{-1} b_2/\bar{\gamma} \\ & 1/\bar{\gamma} \end{pmatrix} = \begin{pmatrix} A_{k-1}^{-1} & -(u_1 + \mu u)/\bar{\gamma} \\ & 1/\bar{\gamma} \end{pmatrix}$$

and

$$\begin{aligned} \bar{A}_k^{-1} \bar{B}_k &= \begin{pmatrix} A_{k-1}^{-1} & -(u_1 + \mu u)/\bar{\gamma} \\ & 1/\bar{\gamma} \end{pmatrix} \begin{pmatrix} b_1 & B \\ \gamma\mu/\rho & \bar{c}_1^T \end{pmatrix} \\ &= \begin{pmatrix} \left(1 - \gamma\mu^2/(\bar{\gamma}\rho)\right) u - (\gamma\mu/(\bar{\gamma}\rho))\, u_1 & A_{k-1}^{-1} B - (u_1 + \mu u)\bar{c}_1^T/\bar{\gamma} \\ \gamma\mu/(\bar{\gamma}\rho) & \bar{c}_1^T/\bar{\gamma} \end{pmatrix}. \end{aligned} \tag{11}$$

Simplifying,

$$1 - \gamma\mu^2/(\bar{\gamma}\rho) = 1 - \mu^2/\rho^2 = \nu^2/\rho^2 \quad \text{and} \quad \gamma\mu/(\bar{\gamma}\rho) = \mu/\rho^2.$$

We also have

$$\begin{aligned} A_{k-1}^{-1} B - (u_1 + \mu u)\bar{c}_1^T/\bar{\gamma} &= U + u c_1^T/\gamma - \mu u \bar{c}_1^T/\bar{\gamma} - u_1 \bar{c}_1^T/\bar{\gamma} \\ &= U + u\,(\rho c_1 - \mu \bar{c}_1)^T/\bar{\gamma} - u_1 \bar{c}_1^T/\bar{\gamma} \\ &= U + \nu u \bar{c}_2^T/\bar{\gamma} - u_1 \bar{c}_1^T/\bar{\gamma}. \end{aligned}$$

Plugging these relations into (11), we get

$$\bar{A}_k^{-1} \bar{B}_k = \begin{pmatrix} \left(\nu^2 u - \mu u_1\right)/\rho^2 & U + \left(\nu u \bar{c}_2^T - u_1 \bar{c}_1^T\right)/\bar{\gamma} \\ \mu/\rho^2 & \bar{c}_1^T/\bar{\gamma} \end{pmatrix}.$$

Let

$$u = \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_{k-1} \end{pmatrix}, \quad u_1 + \mu u = \begin{pmatrix} \bar{\mu}_1 \\ \vdots \\ \bar{\mu}_{k-1} \end{pmatrix}, \quad c_2 = \begin{pmatrix} \nu_2 \\ \vdots \\ \nu_{n-k} \end{pmatrix}, \quad \text{and} \quad \bar{c}_2 = \begin{pmatrix} \bar{\nu}_2 \\ \vdots \\ \bar{\nu}_{n-k} \end{pmatrix}.$$

Then $\omega_*(A_k)$ and $\gamma_*(C_k)$ can be computed from

$$\omega_k(\bar{A}_k) = \bar{\gamma} \quad \text{and} \quad \omega_i(\bar{A}_k)^2 = \omega_i(A_k)^2 + \bar{\mu}_i^2/\bar{\gamma}^2 - \mu_i^2/\gamma^2, \quad 1 \le i \le k-1,$$

and

$$\gamma_1(\bar{C}_k) = \gamma\nu/\rho \quad \text{and} \quad \gamma_j(\bar{C}_k)^2 = \gamma_j(C_k)^2 + \bar{\nu}_j^2 - \nu_j^2, \quad 2 \le j \le n-k.$$

The cost of zeroing out the elements below the diagonal in the $(k+1)$st column is about $4(m-k)(n-k)$ flops, the cost of computing u is about k^2 flops, and the cost of computing $\bar{A}_k^{-1}\bar{B}_k$ is about $4k(n-k)$ flops. Thus the total cost of the modification is about $4m(n-k)+k^2$ flops.

4.4. Efficiency. In this subsection we derive an upper bound on the total number of interchanges and bound the total number of flops. We only consider the case $f > 1$.

Let τ_k be the number of interchanges performed for a particular value of k (i.e., within the inner **while**-loop), and let Δ_k be the determinant of A_k after these interchanges are complete (by convention, $\Delta_0 = 1$). Since $\det(A_k) = \Delta_{k-1}\,\gamma_{j_{\max}}(C_{k-1})$ before the interchanges, and each interchange increases $\det(A_k)$ by at least a factor of f, it follows that

$$\Delta_k \geq \Delta_{k-1}\,\gamma_{j_{\max}}(C_{k-1})\,f^{\tau_k}.$$

By (3), we have

$$\sigma_{l+1}(M) \leq \sigma_{\max}\left(\mathcal{C}_l(M)\right) \leq \|\mathcal{C}_l(M)\|_F \leq \sqrt{n-l}\;\gamma_{j_{\max}}\left(\mathcal{C}_l(M)\right),$$

for $1 \leq l < n$, so that

$$\Delta_k \geq \Delta_{k-1}\,\frac{1}{\sqrt{n}}\,\sigma_k(M)\,f^{\tau_k} \geq \left(\frac{1}{\sqrt{n}}\right)^k \left\{\prod_{i=1}^{k}\sigma_i(M)\right\} f^{t_k},$$

where $t_k = \sum_{i=1}^{k}\tau_i$ is the total number of interchanges up to this point. On the other hand, from (2) we also have

$$\Delta_k = \prod_{i=1}^{k}\sigma_i(A_k) \leq \prod_{i=1}^{k}\sigma_i(M).$$

Combining these relations, we have $f^{t_k} \leq (\sqrt{n})^k$, so that $t_k \leq k\log_f\sqrt{n}$.

The cost of the updating procedure is about $2(2m-k)(n-k)$ flops (see §4.1), the cost of the reduction procedure is at most about $3k(2n-k)$ flops (see §4.2), and the cost of the modifying procedure is about $4m(n-k)+k^2$ flops (see §4.3). For each increase in k and each interchange, the cost of finding $\hat{\rho}(R,k)$ is about $2k(n-k)$ flops (taking $k(n-k)$ absolute values and making $k(n-k)$ comparisons).

Let k_f be the final value of k when Algorithm 5 halts. Then the total number of interchanges t_{k_f} is bounded by $k_f\log_f\sqrt{n}$, which is $O(k_f)$ when f is taken to be a small power of n (e.g., $\sqrt{n}$ or n). Thus the total cost is at most about

$$\begin{aligned}&\sum_{k=1}^{k_f}\left[2(2m-k)(n-k)+2k(n-k)\right]\\ &\qquad + t_{k_f}\max_{1\leq k\leq k_f}\left[3k(2n-k)+4m(n-k)+k^2+2k(n-k)\right]\\ &\leq 2mk_f(2n-k_f)+4t_{k_f}n(m+n)\end{aligned}$$

flops. When f is taken to be a small power of n (e.g., $\sqrt{n}$ or n), the total cost is $O(mnk_f)$ flops. Normally t_{k_f} is quite small (see §6), and thus the cost is about $2mk_f(2n-k_f)$ flops. When $m \gg n$, Algorithm 5 is almost as fast as Algorithm 1; when $m \approx n$, Algorithm 5 is about 50% more expensive. We will discuss efficiency further in §§6 and 8.

5. Numerical stability. Since we update and modify $\omega_*(A_k)$, $\gamma_*(C_k)$, and $A_k^{-1}B_k$ rather than recompute them, we might expect some loss of accuracy. But since we only use these quantities for deciding which pairs of columns to interchange, Algorithm 5 could only be unstable if they were extremely inaccurate.

In §5.1 we give an upper bound for $\rho(R,k)$ during the interchanges. Since this bound grows slowly with k, Theorem 3.2 asserts that A_k can never be extremely ill conditioned, provided that $\sigma_k(M)$ is not very much smaller than $\|M\|_2$. This implies that the elements of $A_k^{-1}B_k$ cannot be too inaccurate. In §5.2 we discuss the numerical stability of updating and modifying $\omega_*(A_k)$ and $\gamma_*(C_k)$.

5.1. An upper bound on $\rho(R,k)$ during interchanges.

We only consider the case $f > 1$.

LEMMA 5.1. *Let $A, C, U \in \mathbf{R}^{k\times k}$, where A is upper triangular with positive diagonal elements and $U = (u_{i,j})$. If*

$$\sqrt{u_{i,j}^2 + \left(\gamma_j(C)/\omega_i(A)\right)^2} \le f, \qquad 1 \le i, j \le k,$$

then

$$\sqrt{\det\left[(AU)^T AU + C^T C\right]} \le \det(A)\,\left(\sqrt{2k}\,f\right)^k.$$

Proof. First, note that

$$\sqrt{\det\left[(AU)^T AU + C^T C\right]} = \prod_{i=1}^{k} \sigma_i\left(\begin{pmatrix} AU \\ C \end{pmatrix}\right).$$

Let $\alpha = \sigma_{\min}(A)$, and write

$$W \equiv \begin{pmatrix} AU \\ C \end{pmatrix} = \begin{pmatrix} A & \\ & \alpha I_k \end{pmatrix}\begin{pmatrix} U \\ C/\alpha \end{pmatrix} \equiv \tilde{D}\,\tilde{W}.$$

By [29, Thm. 3.3.4], we have

$$\prod_{i=1}^{k} \sigma_i(W) \le \prod_{i=1}^{k} \sigma_i(\tilde{D})\,\sigma_i(\tilde{W}).$$

Since $\sigma_i(\tilde{D}) = \sigma_i(A)$, for $1 \le i \le k$, we have

$$\prod_{i=1}^{k} \sigma_i(\tilde{D}) = \prod_{i=1}^{k} \sigma_i(A) = \det(A).$$

Now, since $\tilde{W}^T\tilde{W}$ is symmetric and positive definite,

$$\prod_{i=1}^{k} \sigma_i(\tilde{W}) = \sqrt{\det(\tilde{W}^T\tilde{W})} \le \sqrt{\prod_{i=1}^{k} (\tilde{W}^T\tilde{W})_{i,i}} = \prod_{i=1}^{k} \|\tilde{W}e_i\|_2,$$

and, since

$$\frac{1}{\alpha} = \|A^{-1}\|_2 \le \sqrt{k}\max_{1\le i\le k}\frac{1}{\omega_i(A)} = \frac{\sqrt{k}}{\min\limits_{1\le i\le k}\omega_i(A)},$$

we have

$$\|\tilde{W}e_j\|_2^2 = \sum_{i=1}^{k} u_{ij}^2 + \frac{\gamma_j(C)^2}{\alpha^2} \le kf^2 + \frac{k\gamma_j(C)^2}{\min\limits_{1\le i\le k}\omega_i(A)^2} \le 2kf^2.$$

The result follows immediately. □

To derive an upper bound on $\rho(R,k)$ during the interchanges, we use techniques similar to those used by Wilkinson [43] to bound the growth factor for Gaussian elimination with complete pivoting.[5] Let

$$\mathcal{W}(\tau) = \left(\tau\prod_{s=2}^{\tau} s^{1/(s-1)}\right)^{1/2},$$

[5] See [13] for a connection between the growth factor for Gaussian elimination with *partial* pivoting and the failure of RRQR algorithms.

which is Wilkinson's upper bound on the growth factor for Gaussian elimination with complete pivoting on a $\tau \times \tau$ matrix. Although $\mathcal{W}(\tau)$ is not a polynomial in τ, it grows rather slowly [43]: $\mathcal{W}(\tau) = O\big(\tau^{1+\frac{1}{4}\log\tau}\big)$.

THEOREM 5.2. *If Algorithm 5 performs τ interchanges for some $k > 1$, then*

$$\rho(\mathcal{R}_k(M\,\Pi), k) \le 2\sqrt{6}\, f\, (\tau+1)\, \mathcal{W}(\tau+1).$$

Proof. Assume that Algorithm 5 will perform at least one interchange for this value of k; otherwise the result holds trivially.

Let $\Pi^{(l)}$ be the permutation after the first l interchanges, where $0 \le l \le \tau + 1$. Partition

$$M\,\Pi^{(l)} = \begin{pmatrix} M_k^{(l)} & M_{n-k}^{(l)} \end{pmatrix},$$

where $M_k^{(l)} \in \mathbf{R}^{m\times k}$ and $M_{n-k}^{(l)} \in \mathbf{R}^{m\times(n-k)}$. Assume that $\eta(l,\tau)$ columns of $M_k^{(\tau+1)}$ are from $M_{n-k}^{(l)}$, and that the rest are from $M_k^{(l)}$. Since there are $\tau - l + 1$ more interchanges, we have[6] $\eta(l,\tau) \le \tau - l + 1$.

Without loss of generality, we assume that the first $k - \eta(l,\tau)$ columns of $M_k^{(\tau+1)}$ are the first $k - \eta(l,\tau)$ columns of $M_k^{(l)}$, and that the last $\eta(l,\tau)$ columns of $M_k^{(\tau+1)}$ are the first $\eta(l,\tau)$ columns of $M_{n-k}^{(l)}$. Then we can write

$$R^{(l)} \equiv \mathcal{R}_k(M\,\Pi^{(l)}) \equiv \begin{pmatrix} A_k^{(l)} & B_k^{(l)} \\ & C_k^{(l)} \end{pmatrix} = \begin{pmatrix} A_{1,1} & A_{1,2} & B_{1,1} & B_{1,2} \\ & A_{2,2} & B_{2,1} & B_{2,2} \\ & & C_{1,1} & C_{1,2} \\ & & & C_{2,2} \end{pmatrix},$$

where $A_{2,2}, C_{1,1} \in \mathbf{R}^{\eta(l,\tau)\times\eta(l,\tau)}$ and the partition is such that

$$R^{(\tau+1)} \equiv \mathcal{R}_k(M\,\Pi^{(\tau+1)}) \equiv \begin{pmatrix} A_k^{(\tau+1)} & B_k^{(\tau+1)} \\ & C_k^{(\tau+1)} \end{pmatrix} = \mathcal{R}_k \begin{pmatrix} A_{1,1} & B_{1,1} & A_{1,2} & B_{1,2} \\ & B_{2,1} & A_{2,2} & B_{2,2} \\ & C_{1,1} & & C_{1,2} \\ & & & C_{2,2} \end{pmatrix}.$$

These relations imply that

$$\det(A_k^{(l)}) = \det(A_{1,1})\,\det(A_{2,2}) \tag{12}$$

and

$$\det(A_k^{(\tau+1)}) = \det(A_{1,1})\sqrt{\det\left[B_{2,1}^T B_{2,1} + C_{1,1}^T C_{1,1}\right]}. \tag{13}$$

Let $f^{(l)} = \rho(R^{(l)}, k)$. By the definition of $\rho(R,k)$, we have

$$\sqrt{\left(A_{2,2}^{-1}B_{2,1}\right)_{i,j}^2 + \left(\gamma_j(C_{1,1})/\omega_i(A_{2,2})\right)^2} \le f^{(l)}$$

for $1 \le i, j \le \eta(l,\tau)$. Applying Lemma 5.1 and recalling that $\eta(l,\tau) \le \tau - l + 1$, we have

$$\sqrt{\det\left[B_{2,1}^T B_{2,1} + C_{1,1}^T C_{1,1}\right]} \le \det(A_{2,2})\left(\sqrt{2(\tau - l + 1)}\, f^{(l)}\right)^{\tau-l+1}.$$

Combining with (12) and (13), we get

$$\det(A_k^{(\tau+1)}) \le \det(A_k^{(l)})\left(\sqrt{2(\tau - l + 1)}\, f^{(l)}\right)^{\tau-l+1}.$$

[6] It is possible that $\eta(l,\tau) < \tau - l + 1$ since a column may be interchanged more than once.

On the other hand, Algorithm 5 ensures that

$$\det(A_k^{(\tau+1)}) \geq \det(A_k^{(l)}) \left(f^{(l)}/\sqrt{2}\right) \cdots \left(f^{(\tau)}/\sqrt{2}\right).$$

Comparing these two relations, we have

$$f^{(l)} \cdots f^{(\tau)} \leq \left(2\sqrt{\tau - l + 1}\, f^{(l)}\right)^{\tau - l + 1}, \quad 0 \leq l \leq \tau. \tag{14}$$

Since

$$\sum_{l=1}^{s-1} \frac{1}{(\tau - l)(\tau - l + 1)} + \frac{1}{\tau} = \frac{1}{\tau - s},$$

taking the product of the $(\tau - l)(\tau - l + 1)$st root of (14) with $l = 1, 2, \ldots, \tau - 1$ and the τth root of (14) with $l = 0$, we have

$$\begin{aligned} \prod_{s=0}^{\tau-1} \left(f^{(s)}\right)^{1/(\tau - s)} f^{(\tau)} &\leq \left(\prod_{l=1}^{\tau-1} \left(2\sqrt{\tau - l + 1}\, f^{(l)}\right)^{1/(\tau - l)}\right) \left(2\sqrt{\tau + 1}\, f^{(0)}\right)^{(\tau+1)/\tau} \\ &\leq 2^{1 + \sum_{l=0}^{\tau-1} 1/(\tau - l)} \left((\tau + 1) \prod_{l=0}^{\tau-1} (\tau - l + 1)^{1/(\tau - l)}\right)^{1/2} \left(\prod_{l=0}^{\tau-1} \left(f^{(l)}\right)^{1/(\tau - l)}\right) f^{(0)}, \end{aligned}$$

which simplifies to

$$f^{(\tau)} \leq f^{(0)}\, 2^{1 + \sum_{s=1}^{\tau} 1/s} \left((\tau + 1) \prod_{s=2}^{\tau+1} s^{1/(s-1)}\right)^{1/2} \leq 2 f^{(0)}\, (\tau + 1) \mathcal{W}(\tau + 1).$$

Remark 2 at the end of §4.1 implies that $f^{(0)} \leq \sqrt{6}\, f$. Plugging this into the last relation proves the result. □

From §4.4 we have $\tau_k \leq k \log_f \sqrt{n}$. For example, when $\sqrt{n} \leq f \leq n$, we have $\tau_k \leq k$, so that $\rho(R, k) \leq O\left(n\, k\, \mathcal{W}(k)\right)$.

5.2. Computing the row norms of A_k^{-1} and the column norms of C_k. In this section we discuss the numerical stability of updating and modifying $\omega_*(A_k)$ and $\gamma_*(C_k)$ as a result of interchanges, assuming that f is a small power of n.

For any $\alpha > 0$, we let $\mathcal{O}_n(\alpha)$ denote a positive number that is bounded by α times a slowly increasing function of n. By Theorems 3.2 and 5.2, $\|A_k^{-1}\|_2 = \mathcal{O}_n\left(1/\sigma_k(M)\right)$ and $\|C_k\|_2 = \mathcal{O}_n\left(\sigma_{k+1}(M)\right)$ after each interchange. As Algorithm 5 progresses, $\|A_k^{-1}\|_2$ increases from $\mathcal{O}_n\left(1/\sigma_1(M)\right)$ to $\mathcal{O}_n\left(1/\sigma_k(M)\right)$, while $\|C_k\|_2$ decreases from $\mathcal{O}_n\left(\sigma_1(M)\right)$ to $\mathcal{O}_n\left(\sigma_{k+1}(M)\right)$. A straightforward error analysis shows that the errors in $1/\omega_i(A_k)^2$ and $\gamma_j(C_k)^2$ are bounded by $\mathcal{O}_n\left(\epsilon/\sigma_k^2(M)\right)$ and $\mathcal{O}_n\left(\epsilon\, \sigma_1^2(M)\right)$, respectively, where ϵ is the machine precision. Hence the error in $1/\omega_i(A_k)^2$ is less serious than the error in $\gamma_j(C_k)^2$, which can be larger than $\|C_k\|_2^2$ when $\|C_k\|_2 \leq \mathcal{O}_n\left(\sqrt{\epsilon}\, \sigma_1(M)\right)$.

Algorithm 5 uses the computed values of $\omega_*(A_k)$ and $\gamma_*(C_k)$ only to decide which columns to interchange. But although these values do not need to be very accurate, we do need to avoid the situation where they have no accuracy at all. Thus we recompute rather than update or modify $\gamma_*(C_k)$ when $\max_{1 \leq j \leq n-k} \gamma_j(C_k) = \mathcal{O}_n\left(\sqrt{\epsilon}\, \sigma_1(M)\right)$. This needs to be done at most twice if one wants to compute a strong RRQR factorization with A_k numerically nonsingular. A similar approach is taken in xgeqpf, the LAPACK [1] implementation of Algorithm 1.

6. Numerical experiments. In this section we report some numerical results for a Fortran implementation (SRRQR) of Algorithm 5 and the all-Fortran implementation (DGEQPF) of Algorithm 1 in LAPACK [1]. The computations were done on a SPARCstation/10 in double precision where the machine precision is $\epsilon = 1.1 \times 10^{-16}$.

We use the following sets of $n \times n$ test matrices:

1. *Random*: a random matrix with elements uniformly distributed in $[-1, 1]$;
2. *Scaled random*: a random matrix whose ith row is scaled by the factor $\eta^{i/n}$, where $\eta > 0$;
3. *GKS*: an upper-triangular matrix whose jth diagonal element is $1/\sqrt{j}$ and whose (i, j) element is $-1/\sqrt{j}$, for $j > i$ (see Golub, Klema, and Stewart [22]);
4. *Kahan* (see Example 1 in §2);
5. *Extended Kahan*: the matrix $M = S_{3l} R_{3l}$, where
$$S_{3l} = \mathrm{diag}(1, \varsigma, \varsigma^2, \ldots, \varsigma^{3l-1}) \quad \text{and} \quad R_{3l} = \begin{pmatrix} I_l & -\varphi H_l & 0 \\ & I_l & \varphi H_l \\ & & \mu I_l \end{pmatrix};$$
l is a power of 2; $\varsigma > 0$, $\varphi > 1\big/\sqrt[4]{4l-1}$, and $\varsigma^2 + \varphi^2 = 1$; $0 < \mu \ll 1$; and $H_l \in \mathbf{R}^{l \times l}$ is a symmetric Hadamard matrix (i.e., $H_l^2 = l\, I_l$ and every component of H_l is ± 1).

In particular, we chose $\eta = 20\epsilon$, $\varphi = 0.285$, and $\mu = 20\epsilon/\sqrt{n}$.

In exact arithmetic Algorithm 1 does not perform any interchanges for the Kahan and extended Kahan matrices. To preserve this behavior in DGEQPF we scaled the jth columns of these matrices by $1 - 100 j\sqrt{\epsilon}$ and $1 - 10 j\epsilon$, respectively, for $1 \le j \le n$. To prevent DGEQPF from taking advantage of the upper-triangular structure we replaced all of the zero entries by random numbers of the order ϵ^2.

For each test matrix, we took $n = 96$, 192, and 384, and set $f = 10\sqrt{n}$ and $\delta \approx 3 \times 10^{-13} \times \|M\|_2$ in SRRQR. For the extended Kahan matrix, we also used $f = \varphi^2 l 1$ and $\delta = 4l^2 \sigma_{2l+1}(M)$; these results are labeled *Extended Kahan**.

The results are summarized in Tables 1 and 2. Execution time is in seconds; rank is the value of k computed by SRRQR; t_{k_f} is the total number of interchanges in the inner **while**-loop of SRRQR; and
$$q_1(k, n) = \sqrt{1 + 2f^2 k(n-k)} \quad \text{and} \quad q_2(k, n) = \begin{cases} f & \text{if } k < n \\ 0 & \text{if } k = n \end{cases}$$
are the theoretical upper bounds on
$$\max_{1 \le i \le k,\ 1 \le j \le n-k} \left(\frac{\sigma_i(M)}{\sigma_i(A_k)}, \frac{\sigma_j(C_k)}{\sigma_{k+j}(M)} \right) \quad \text{and} \quad \max_{1 \le i \le k,\ 1 \le j \le n-k} \left| \left(A_k^{-1} B_k\right)_{i,j} \right|,$$
respectively, for SRRQR.

The execution times confirm that Algorithm 5 is about 50% more expensive than Algorithm 1 on matrices that require only a small number t_{k_f} of interchanges. And as predicted, Algorithm 1 failed to reveal the numerical rank of the Kahan matrix. Finally, the results suggest that the theoretical upper bounds $q_1(k, n)$ and $q_2(k, n)$ are much too large for $0 < k < n$.

For the extended Kahan matrices with $f = \varphi^2 l$ there were no interchanges until the $2l$th step, when the ith column was interchanged with the $(2l + i)$th column for $i = 1, 2, \ldots, l$. These $l = n/3$ column interchanges show that Algorithm 5 may have to perform $O(n)$ interchanges before finding a strong RRQR factorization for a given f (see §4.4) and can be more than twice as expensive as Algorithm 1. However, the extended Kahan matrix is already a strong RRQR factorization with $f = 10\sqrt{n}$ for the values of n used here, which is why no interchanges were necessary.

TABLE 1
SRRQR versus DGEQPF: Execution time.

Matrix type	Order N	Execution time		Rank k	t_{k_f}
		SRRQR	DGEQPF		
Random	96	0.20	0.13	96	0
	192	1.57	0.98	192	0
	384	16.2	11.0	384	0
Scaled random	96	0.19	0.15	74	0
	192	1.48	1.16	147	0
	384	14.5	11.4	290	0
GKS	96	0.20	0.13	95	0
	192	1.58	1.00	191	0
	384	15.5	10.7	383	0
Kahan	96	0.21	0.13	95	1
	192	1.59	0.99	191	1
	384	15.7	10.5	383	1
Extended Kahan	96	0.17	0.15	64	0
	192	1.38	1.16	128	0
	384	13.4	11.4	256	0
Extended Kahan*	96	0.38	0.15	64	32
	192	3.21	1.16	128	64
	384	31.4	11.7	256	128

TABLE 2
SRRQR versus DGEQPF: Bounds.

Matrix type	Order N	$\max_{i,j}\left(\frac{\sigma_i(M)}{\sigma_i(A_k)}, \frac{\sigma_j(C_k)}{\sigma_{k+j}(M)}\right)$			$\max_{i,j}\left\lvert\left(A_k^{-1}B_k\right)_{i,j}\right\rvert$		
		SRRQR	$q_1(k,n)$	DGEQPF	SRRQR	$q_2(k,n)$	DGEQPF
Random	96	1	1	1	0	0	0
	192	1	1	1	0	0	0
	384	1	1	1	0	0	0
Scaled random	96	2.93	5.59×10^3	2.38	1.71	98.0	1.75
	192	3.39	1.13×10^4	3.04	1.58	1.96×10^2	1.41
	384	3.75	2.29×10^4	3.76	1.37	3.92×10^2	1.16
GKS	96	1.12	1.35×10^3	1.12	0.71	98.0	0.71
	192	1.09	1.91×10^3	1.09	0.71	1.96×10^2	0.71
	384	1.07	2.71×10^3	1.07	0.71	3.92×10^2	0.71
Kahan	96	1.04	1.35×10^3	1.04×10^{10}	0.78	98.0	4.92×10^9
	192	1.04	1.91×10^3	1.86×10^{17}	0.78	1.96×10^2	1.40×10^{20}
	384	1.04	2.71×10^3	5.98×10^{16}	0.78	3.92×10^2	1.27×10^{23}
Extended Kahan	96	3.22	6.27×10^3	3.22	2.60	98.0	2.60
	192	5.76	1.25×10^4	5.76	5.20	1.96×10^2	5.20
	384	10.9	2.51×10^4	10.9	10.4	3.92×10^2	10.4
Extended Kahan*	96	1.17	1.66×10^2	3.22	0.38	2.60	2.60
	192	1.09	6.65×10^2	5.76	0.19	5.20	5.20
	384	1.05	2.66×10^3	10.9	0.10	10.4	10.4

7. Algorithm 1 and the strong RRQR factorization.

Using the techniques developed in §3, we now show that Algorithm 1 satisfies (5) and (6) with $q_1(k, n)$ and $q_2(k, n)$ functions that grow exponentially with k. We need the following lemma.

LEMMA 7.1 (Faddeev, Kublanovskaya, and Faddeeva [16]). *Let $W = (w_{i,j}) \in \mathbf{R}^{n\times n}$ be an upper-triangular matrix with $w_{i,i} = 1$ and $|w_{i,j}| \le 1$ for $1 \le i \le j \le n$. Then*

$$|(W^{-1})_{i,j}| \le 2^{n-2}, \quad 1 \le i, j \le n, \qquad \textit{and} \qquad \|W^{-1}\|_F \le \frac{\sqrt{4^n + 6n - 1}}{3}.$$

THEOREM 7.2. *Let Π be the permutation chosen by Algorithm* 1, *and let*

$$R \equiv \begin{pmatrix} A_k & B_k \\ & C_k \end{pmatrix} = \mathcal{R}_k(M\,\Pi).$$

Then

$$\sigma_i(A_k) \geq \frac{\sigma_i(M)}{\sqrt{n-i}\,2^i}, \tag{15}$$

$$\sigma_j(C_k) \leq \sigma_{k+j}(M)\,\sqrt{n-k}\,2^k, \tag{16}$$

and

$$\left|\left(A_k^{-1}B_k\right)_{i,j}\right| \leq 2^{k-i}, \tag{17}$$

for $1 \leq i \leq k$ *and* $1 \leq j \leq n-k$.

Proof. For simplicity we assume that M (and therefore R) has full rank.

Let

$$R \equiv \begin{pmatrix} A_k & B_k \\ & C_k \end{pmatrix} = D \begin{pmatrix} W_{1,1} & W_{1,2} \\ & W_{2,2} \end{pmatrix} \equiv DW \quad \text{and} \quad \widetilde{W}_j = \begin{pmatrix} W_{1,1} & w_j \\ & 1 \end{pmatrix},$$

where $D = \operatorname{diag}(d_1, d_2, \ldots, d_m)$ is the diagonal of R, $W_{1,1} \in \mathbf{R}^{k\times k}$ is unit upper triangular, $W_{1,2} \in \mathbf{R}^{k\times(n-k)}$, $W_{2,2} \in \mathbf{R}^{(m-k)\times(n-k)}$, and $w_j \in \mathbf{R}^k$ is the jth column of $W_{1,2}$. Since Algorithm 1 would not cause any column interchanges if it were applied to R, it follows that $d_1 \geq d_2 \geq \cdots \geq d_k$ and that no component of $\widetilde{W}_j$ has absolute value larger than 1.

Let $u_{i,j} = \left(A_k^{-1}B_k\right)_{i,j}$. Then $-u_{i,j}$ is the $(i, k+1)$ component of $\widetilde{W}_j^{-1}$. Applying the first result in Lemma 7.1 to the lower right $(k-i+2) \times (k-i+2)$ submatrix of $\widetilde{W}_j$, we have $|u_{i,j}| \leq 2^{k-i}$, which is (17).

As in the proof of Theorem 3.2, let $\alpha = \sigma_{\max}(C_k)/\sigma_{\min}(A_k)$, and write

$$\tilde{R}_2 = \begin{pmatrix} \alpha A_k & \\ & C_k \end{pmatrix} = \begin{pmatrix} A_k & B_k \\ & C_k \end{pmatrix} \begin{pmatrix} \alpha I_k & -A_k^{-1}B_k \\ & I_{n-k} \end{pmatrix} \equiv RW_2.$$

Then

$$\sigma_j(C_k) = \sigma_{j+k}(\tilde{R}_2) \leq \sigma_{j+k}(R)\,\|W_2\|_2 = \sigma_{j+k}(M)\,\|W_2\|_2.$$

But

$$\begin{aligned} \|W_2\|_2^2 &\leq 1 + \|A_k^{-1}B_k\|_2^2 + \alpha^2 \\ &\leq 1 + \sum_{i=1}^{k}\sum_{j=1}^{n-k} u_{i,j}^2 + \|C_k\|_F^2 \|A_k^{-1}\|_F^2 \\ &= 1 + \sum_{i=1}^{k}\sum_{j=1}^{n-k} \{u_{i,j}^2 + \left(\gamma_j(C_k)/\omega_i(A_k)\right)^2\}. \end{aligned}$$

Since $1/\omega_i(A_k) \leq 1/(d_k\omega_i(W_{1,1}))$ and $\gamma_j(C_k) \leq d_k$, we have

$$u_{i,j}^2 + \left(\gamma_j(C_k)/\omega_i(A_k)\right)^2 \leq (\widetilde{W}_j^{-1})_{i,k+1}^2 + 1/\omega_i(W_{1,1})^2 = 1/\omega_i(\widetilde{W}_j)^2.$$

Using the second result in Lemma 7.1, it follows that

$$\sum_{i=1}^{k} \left\{u_{i,j}^2 + \left(\gamma_j(C_k)/\omega_i(A_k)\right)^2\right\} \leq \sum_{i=1}^{k} 1/\omega_i(\widetilde{W}_j)^2 = \|\widetilde{W}_j^{-1}\|_F^2 - 1 \leq 4^k - 1,$$

so that $\|W_2\|_2^2 \leq 4^k(n-k)$, which gives (16).

Similarly, writing

$$R = \begin{pmatrix} A_k & \\ & C_k/\alpha \end{pmatrix} \begin{pmatrix} I_k & A_k^{-1}B_k \\ & \alpha I_{n-k} \end{pmatrix} \equiv \tilde{R}_1 W_1,$$

we have

$$\sigma_k(M) = \sigma_k(R) \le \sigma_k(\tilde{R}_1) \, \|W_1\|_2 \le \sigma_k(A_k) \, \sqrt{n-k} \, 2^k.$$

Taking $k = i$ and noting that $\sigma_i(A_i) \le \sigma_i(A_k)$ by the interlacing property of the singular values [24, Cor. 8.3.3], we get (15). □

If R has very few linearly independent columns, then we can stop Algorithm 1 with a small value of k and are guaranteed to have a strong RRQR factorization. Results similar to Theorem 7.2 can be obtained for the RRQR algorithms in [10, 18, 25], and [39].

8. Some extensions. We have proved the existence of a strong RRQR factorization for a matrix $M \in \mathbf{R}^{m\times n}$ with $m \ge n$ and presented an efficient algorithm for computing it. In this section, we describe some further improvements and extensions of these results.

Since Algorithm 1 seems to work well in practice [5, 10, 11, 13], Algorithm 5 tends to perform very few (and quite often no) interchanges in its inner **while**-loop. This suggests using Algorithm 1 as an initial phase (cf. [13] and [37]), and then using Algorithm 4 to remove any dependent columns from A_k, reducing k as needed (cf. [10] and [18]). In many respects the resulting algorithm is equivalent to applying Algorithm 5 to M^{-1} (cf. Stewart [39]).

ALGORITHM 6. Compute k and a strong RRQR factorization.

```
$k := 0$;  $R \equiv C_k := M$;  $\Pi := I$;
Compute $\gamma_*(C_k)$;
while $\max_{1\le j\le n-k} \gamma_j(C_k) \ge \delta$ do
  $j_{\max} := \operatorname{argmax}_{1\le j\le n-k} \gamma_j(C_k)$;
  $k := k+1$;
  Compute $R \equiv \begin{pmatrix} A_k & B_k \\ & C_k \end{pmatrix} := \mathcal{R}_k(R\,\Pi_{k,k+j_{\max}-1})$ and $\Pi := \Pi\,\Pi_{k,k+j_{\max}-1}$;
  Update $\gamma_*(C_k)$;
endwhile;
Compute $\omega_*(A_k)$ and $A_k^{-1}B_k$;
repeat
  while $\hat{\rho}(R,k) > f$ do
    Find $i$ and $j$ such that $\left|(A_k^{-1}B_k)_{i,j}\right| > f$ or $\gamma_j(C_k)/\omega_i(A_k) > f$;
    Compute $R \equiv \begin{pmatrix} A_k & B_k \\ & C_k \end{pmatrix} := \mathcal{R}_k(R\,\Pi_{i,j+k})$ and $\Pi := \Pi\,\Pi_{i,j+k}$;
    Modify $\omega_*(A_k)$, $\gamma_*(C_k)$, and $A_k^{-1}B_k$;
  endwhile;
  if $\min_{1\le i\le k} \omega_i(A_k) \le \delta$ then
    $i_{\min} := \operatorname{argmin}_{1\le i\le k} \omega_i(A_k)$;
    $k := k-1$;
    Compute $R \equiv \begin{pmatrix} A_k & B_k \\ & C_k \end{pmatrix} := \mathcal{R}_k(R\,\Pi_{i_{\max},k+1})$ and $\Pi := \Pi\,\Pi_{i_{\max},k+1}$;
    Downdate $\omega_*(A_k)$, $\gamma_*(C_k)$, and $A_k^{-1}B_k$;
  endif;
until $k$ is unchanged;
```

As before, Algorithm 6 eventually halts and finds k and a strong RRQR factorization. The total number of interchanges t_k is bounded by $(n-k)\log_f \sqrt{n}$, which is $O(n-k)$ when f is taken to be a small power of n (see §4.4). The formulas for downdating $\omega_*(A_k)$, $\gamma_*(C_k)$, and $A_k^{-1}B_k$ are analogous to those in §4.1.

Algorithm 6 initializes $\omega_*(A_k)$ and $A_k^{-1}B_k$ after the first **while**-loop, at a cost of $O(k^2n)$ flops. However, since they are only used to decide which (if any) columns to interchange and whether to decrease k, they do not need to be computed very accurately. To make the algorithm more efficient, we could instead use the condition estimation techniques in [4, 5, 10, 27], and [40] to generate unit vectors u and v such that

$$\|A_k^{-1}u\|_2 \approx \|A_k^{-1}\|_2 \quad \text{and} \quad \|B_k^T A_k^{-T} v\|_2 \approx \|B_k^T A_k^{-T}\|_2 \,.$$

Let the $i_{\max}$th component of $A_k^{-1}u$ be the largest in absolute value. To find the smallest entry in $\omega_*(A_k)$, we note that

$$1/\omega_{i_{\max}}(A_k) \approx \max_{1\le i\le k} 1/\omega_i(A_k) \approx \left|(A_k^{-1}u)_{i_{\max}}\right| \,.$$

Similarly, let the $j_{\max}$th component of $B_k^T A_k^{-T} v$ be the largest in absolute value. To find the largest entry of $A_k^{-1}B_k$ in absolute value, we compute the $j_{\max}$th column of $A_k^{-1}B_k$ and look for the largest component in absolute value. Since the condition estimates cost $O(n^2)$ flops, the resulting algorithm will take nearly the same number of flops as QR with column pivoting when at most a few interchanges are needed. As Algorithm 6 could take $O(n)$ interchanges and all condition estimation techniques can fail, Algorithm 6 could be very inefficient and can fail as well, although we believe that this is quite unlikely in practical applications.

Most of the floating-point operations in Algorithms 5 and 6 can be expressed as Level-2 BLAS. Using ideas similar to those in [3] and [6], it should be straightforward to develop block versions of these algorithms so that most of the floating-point operations are performed as Level-3 BLAS.

The restriction $m \ge n$ is not essential and can be removed with minor modifications to Algorithms 5 and 6. Thus these algorithms can also be used to compute a strong RRQR factorization for M^T, which may be preferable when one wants to compute an orthogonal basis for the right approximate null space.

Finally, the techniques developed in this paper can easily be adopted to compute *rank-revealing* LU factorizations [9, 13, 31, 32]. This result will be reported at a later date.

Acknowledgments. The authors thank Shivkumar Chandrasekaran and Ilse Ipsen for many helpful discussions and suggestions, and Gene Golub, Per Christian Hansen, W. Kahan, and Pete Stewart for suggestions that improved the presentation.

REFERENCES

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK Users' Guide*, 2nd ed., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1994.

[2] T. Beelen and P. Van Dooren, *An improved algorithm for the computation of Kronecker's canonical form of a singular pencil*, Linear Algebra Appl., 105 (1988), pp. 9–65.

[3] C. H. Bischof, *A block QR factorization algorithm using restricted pivoting*, in Proceedings, Supercomputing '89, ACM Press, New York, 1989, pp. 248–256.

[4] ———, *Incremental condition estimation*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 312–322.

[5] C. H. Bischof and P. C. Hansen, *Structure preserving and rank-revealing QR-factorizations*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 1332–1350.

[6] ———, *A block algorithm for computing rank-revealing QR factorizations*, Numerical Algorithms, 2 (1992), pp. 371–392.

[7] C. H. Bischof and G. M. Shroff, *On updating signal subspaces*, IEEE Trans. Signal Processing, 40 (1992), pp. 96–105.

[8] P. A. Businger and G. H. Golub, *Linear least squares solutions by Householder transformations*, Numer. Math., 7 (1965), pp. 269–276.

[9] T. F. Chan, *On the existence and computation of LU-factorizations with small pivots*, Math. Comp., 42 (1984), pp. 535–547.

[10] ———, *Rank revealing QR factorizations*, Linear Algebra Appl., 88/89 (1987), pp. 67–82.

[11] T. F. Chan and P. C. Hansen, *Computing truncated singular value decomposition least squares solutions by rank revealing QR-factorizations*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 519–530.

[12] ———, *Some applications of the rank revealing QR factorization*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 727–741.

[13] S. Chandrasekaran and I. Ipsen, *On rank-revealing QR factorisations*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 592–622.

[14] ———, *Analysis of a QR algorithm for computing singular values*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 520–535.

[15] J. Demmel, M. Heath, and H. van der Vorst, *Parallel numerical linear algebra*, in Acta Numerica, A. Iserles, ed., Vol. 2, Cambridge University Press, Cambridge, 1993, pp. 111–197.

[16] D. K. Faddeev, V. N. Kublanovskaya, and V. N. Faddeeva, *Solution of linear algebraic systems with rectangular matrices*, Proc. Steklov Inst. Math., 96 (1968), pp. 93–111.

[17] ———, *Sur les systemes lineaires algebriques de matrices rectangulaires et malconditionees*, in Programmation en Mathematiques Numeriques, Editions Centre Nat. Recherche Sci., Paris VII, 1968, pp. 161–170.

[18] L. V. Foster, *Rank and null space calculations using matrix decomposition without column interchanges*, Linear Algebra Appl., 74 (1986), pp. 47–71.

[19] P. E. Gill, G. H. Golub, W. Murray, and M. A. Saunders, *Methods for modifying matrix factorizations*, Math. Comp., 28 (1974), pp. 505–535.

[20] G. H. Golub, *Numerical methods for solving linear least squares problems*, Numer. Math., 7 (1965), pp. 206–216.

[21] ———, *Matrix decompositions and statistical computation*, in Statistical Computation, R. C. Milton and J. A. Nelder, eds., Academic Press, New York, 1969, pp. 365–397.

[22] G. H. Golub, V. Klema, and G. W. Stewart, *Rank Degeneracy and Least Squares Problems*, Tech. Report TR–456, Dept. of Computer Science, University of Maryland, College Park, MD, 1976.

[23] G. H. Golub and V. Pereyra, *The differentiation of pseudo-inverses, separable nonlinear least squares problems and other tales*, in Generalized Inverses and Applications, M. Z. Nashed, ed., Academic Press, New York, 1976, pp. 303–324.

[24] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 2nd ed., The Johns Hopkins University Press, Baltimore, MD, 1989.

[25] W. B. Gragg and G. W. Stewart, *A stable variant of the secant method for solving nonlinear equations*, SIAM J. Numer. Anal., 13 (1976), pp. 880–903.

[26] M. Gu, *Finding Well-Conditioned Similarities to Block-Diagonalize Nonsymmetric Matrices is NP-Hard*, J. of Complexity, 11 (1995), pp. 377–391.

[27] W. W. Hager, *Condition estimates*, SIAM J. Sci. Statist. Comput., 5 (1984), pp. 311–316.

[28] Y. P. Hong and C.-T. Pan, *Rank-revealing QR factorizations and the singular value decomposition*, Math. Comp., 58 (1992), pp. 213–232.

[29] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, Cambridge, 1991.

[30] T.-M. Hwang, W.-W. Lin, and D. Pierce, *An Alternative Column Selection Criterion for a Rank-Revealing QR Factorization*, Tech. Report BCSTECH-93-021, Boeing Computer Services, Seattle, WA, July 1993. To appear in *Math. Comp.*.

[31] ———, *Improved Bound for Rank Revealing LU Factorizations*, Tech. Report BCSTECH-93-007, Boeing Computer Services, Seattle, WA, Oct. 1993.

[32] T.-M. Hwang, W.-W. Lin, and E. K. Yang, *Rank revealing LU factorizations*, Linear Algebra Appl., 175 (1992), pp. 115–141.

[33] W. Kahan, *Numerical linear algebra*, Canad. Math. Bull., 9 (1966), pp. 757–801.

[34] V. E. Kane, R. C. Ward, and G. J. Davis, *Assessment of linear dependencies in multivariate data*, SIAM J. Sci. Statist. Comput., 6 (1985), pp. 1022–1032.

[35] V. N. Kublanovskaya, *AB-Algorithm and its modifications for the spectral problems of linear pencils of matrices*, Numer. Math., 43 (1984), pp. 329–342.

[36] C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*, Prentice-Hall, Englewood Cliffs, NJ, 1974.

[37] C.-T. Pan and P. T. P. Tang, *Bounds on Singular Values Revealed by QR Factorizations*, Preprint MCS-P332-1092, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, Oct. 1992.

[38] G. W. Stewart, *Introduction to Matrix Computations*, Academic Press, New York, 1973.

[39] ———, *Rank degeneracy*, SIAM J. Sci. Statist. Comput., 5 (1984), pp. 403–413.

[40] G. W. Stewart, *Incremental Condition Calculation and Column Selection*, Tech. Report CS TR–2495, Dept. of Computer Science, University of Maryland, College Park, MD, July 1990.

[41] ———, *An updating algorithm for subspace tracking*, IEEE Trans. Signal Processing, 40 (1992), pp. 1535–1541.

[42] ———, *Updating a rank-revealing ULV decomposition*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 494–499.

[43] J. H. Wilkinson, *Error analysis of direct methods of matrix inversion*, J. Assoc. Comput. Mach., 8 (1961), pp. 281–330.

[44] S. Wold, A. Ruhe, H. Wold, and W. J. Dunn, III, *The collinearity problem in linear regression. The partial least squares (PLS) approach to generalized inverses*, SIAM J. Sci. Statist. Comput., 5 (1984), pp. 735–743.

SIAM J. SCI. COMPUT.
Vol. 17, No. 4, pp. 870–883, July 1996

005

PARALLELIZING THE QR ALGORITHM FOR THE UNSYMMETRIC ALGEBRAIC EIGENVALUE PROBLEM: MYTHS AND REALITY*

GREG HENRY† AND ROBERT VAN DE GEIJN‡

Abstract. Over the last few years, it has been suggested that the popular QR algorithm for the unsymmetric Schur decomposition does not parallelize. In this paper, we present both positive and negative results on this subject. In theory, asymptotically perfect speedup can be obtained. In practice, reasonable speedup can be obtained on an MIMD distributed memory computer for a relatively small number of processors. However, we also show theoretically that it is impossible for the standard QR algorithm to be scalable. Performance of a parallel implementation of the LAPACK DLAHQR routine on the Intel Paragon™ system is reported.

Key words. parallel computing, eigenvalue, Schur decomposition, QR algorithm

AMS subject classifications. 65F15, 15A18

1. Introduction.

Distributed memory parallel algorithms for the unsymmetric eigenvalue problem have been elusive. There are several matrix multiply-based methods currently being studied. Auslander and Tsao [2] and Lederman, Tsao, and Turnbull [27] have a matrix multiply-based parallel algorithm, which uses a polynomial mapping of the eigenvalues. Bai and Demmel [4] have another parallel algorithm based on bisection with the matrix sign function. Matrix tearing methods for finding the eigensystem of an unsymmetric Hessenberg matrix have been proposed by Dongarra and Sidani [11]. These involve making a rank-one change to the Hessenberg matrix to create two separate submatrices and then finding the eigenpairs of each submatrix, followed by performing Newton's method on these values to find the solution to the original problem. None of the above algorithms are particularly competitive in the serial case. They suffer from requiring more floating point operations (flops) and/or yield a loss of accuracy when compared with efficient implementations of the QR algorithm.

Efficient sequential (and shared memory parallel) implementations of the QR algorithm use a blocked version of the Francis double implicit shifted algorithm [15] or a variant thereof [23]. There have also been attempts at improving data reuse by increasing the number of shifts either by using a multi-implicit shifted QR algorithm [3] or pipelining several double shifts simultaneously [34, 35].

A number of attempts at parallelizing the QR algorithm have been made. (See Boley and Maier [9], Geist et al. [17], Geist and Davis [16], and Stewart [30].) Distributing the work evenly amongst the processors has proven difficult for conventional storage schemes, especially when compared with the parallel solution of dense linear systems [25, 26]. Communication also becomes a more significant bottleneck for the parallel QR algorithm. As noted by van de Geijn [32] and van de Geijn and Hudson [33], the use of a block Hankel-wrapped storage scheme can alleviate some of the problems involved in parallelizing the QR algorithm.

In this paper, we present a number of results of theoretical significance on the subject. We reexamine the results on the Hankel-wrapped storage schemes in the setting of a parallel implementation of a state-of-the-art sequential implementation. Theoretically we can show that under certain conditions the described approach is asymptotically 100% efficient: if the number of processors is fixed and the problem size grows arbitrarily large, perfect speedup can be approached. However, we also show that our approach is not scalable in the following sense: to maintain a given level of efficiency, the dimension of the matrix must grow

*Received by the editors March 1, 1995; accepted for publication September 12, 1995.

†Intel Supercomputer Systems Division, 14924 N.W. Greenbrier Parkway, Beaverton, OR 97006 (henry@ssd.intel.com).

‡Department of Computer Sciences, The University of Texas, Austin, TX 78712 (rvdg@cs.utexas.edu).

linearly with the number of processors. As a result, it will be impossible to maintain performance as processors are added, since memory requirements grow with the square of the dimension, and physical memory grows only with the number of processors. While this could be a deficiency attributable to our implementation, we also show that for the standard implementations of the sequential QR algorithm, it is impossible to find an implementation with better scalability properties. Finally, we show that these techniques can indeed be incorporated into a real code by giving details of a prototype distributed memory implementation of the serial algorithm DLAHQR [1], the LAPACK version of the double implicit shifted QR algorithm. Full functionality of the LAPACK code can be supported. That is, the techniques can be extended to allow for the cases of computing the Schur vectors, computing the Schur decomposition of H, or just computing the eigenvalues alone. We have implemented a subset of this functionality for which the code is described and performance results are given.

Thus this paper makes four contributions: it describes a data decomposition that allows, at least conceptually, straightforward implementation of the QR algorithm; it gives theoretical limitations for parallelizing the standard QR algorithm; it describes a parallel implementation based on the proposed techniques; it reports performance results of this proof-of-concept implementation obtained on the Intel Paragon™ system.

2. Sequential QR algorithm.

While we assume the reader of this paper to be fully versed in the intricate details of the QR algorithm, we briefly review the basics in this section.

The Francis double implicit shifted QR algorithm has been a successful serial method for computing the Schur decomposition $H = QTQ^T$. Here T is an upper pseudotriangular matrix, with 1×1 or 2×2 blocks along the diagonal, and Q is orthogonal. We assume for simplicity that our initial matrix H is Hessenberg. The parallelization of the reduction to Hessenberg form is a well-understood problem, and, unlike the eigenvalue problem, the Hessenberg reduction has been shown to parallelize well [6, 12].

One step of the Francis double shift Schur decomposition is in Figure 1. Here, the Householder matrices are symmetric orthogonal transforms of the form

$$P_i = I - 2\frac{vv^T}{v^T v},$$

where $v \in \Re^n$ and

$$v_j = \begin{cases} 0 & \text{if } j < i+1 \text{ or } j > i+3, \\ 1 & \text{if } j = i+1. \end{cases}$$

We assume the Hessenberg matrix is unreduced and, if not, we find the largest unreduced submatrix of H. Suppose this submatrix is $H(k : l, k : l)$. We then apply the Francis Hessenberg QR (HQR) step to the rows and columns of H corresponding to the submatrix; that is,

$$\begin{aligned} H(k:l,:) &\leftarrow P_i H(k:l,:), \\ H(:,k:l) &\leftarrow H(:,k:l)P_i. \end{aligned}$$

The double implicit shifts in this case are chosen to be the eigenvalues of

$$e = \text{eig}(H(l-1:l, l-1:l)).$$

In practice, after every couple of iterations, some of the subdiagonals of H will become numerically zero, and at this point the problem deflates into smaller problems.

```
Francis HQR Step
$e = \mathrm{eig}(H(n-1:n, n-1:n))$
Let $x = (H - e(1)I_n) * (H - e(2)I_n)e_1$
Let $P_0 \in \Re^{n \times n}$ be a Householder matrix s.t.
    $P_0 x$ is a multiple of $e_1$.
$H \leftarrow P_0 H P_0$
for $i = 1, \ldots, n-2$
        Compute $P_i$ so that
            $P_i H$ has zero $(i+2, i)$ and
                $(i+3, i)$ entries.
        Update $H \leftarrow P_i H P_i$
        Update $Q \leftarrow Q P_i$
endfor
```

FIG. 1. *Sequential Francis HQR step.*

3. Parallel QR algorithms.

3.1. Data decomposition. We briefly describe the storage scheme presented in [33]. Suppose $A \in \Re^{n \times n}$ where $n = mhp$ for some integers h and m, and p is the number of processors. Suppose A is partitioned as follows:

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,mp-1} & A_{1,mp} \\ A_{2,1} & A_{2,2} & \cdots & A_{2,mp-1} & A_{2,mp} \\ A_{3,1} & A_{3,2} & \cdots & A_{3,mp-1} & A_{3,mp} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{mp,1} & A_{mp,2} & \cdots & A_{mp,mp-1} & A_{mp,mp} \end{bmatrix},$$

where $A_{i,j} \in \Re^{h \times h}$. We denote processor k owning block $A_{i,j}$ with a superscript $A_{ij}^{(k)}$. The one-dimensional block Hankel-wrapped storage for $m = 1$ assigns submatrix $A_{i,j}$ to processor

$$(i + j - 2) \ \mathbf{mod}\ p.$$

That is, the distribution amongst the processors is as follows [33]:

$$\begin{bmatrix} A_{1,1}^{(0)} & A_{1,2}^{(1)} & A_{1,3}^{(2)} & \cdots & A_{1,mp-1}^{(p-2)} & A_{1,mp}^{(p-1)} \\ A_{2,1}^{(1)} & A_{2,2}^{(2)} & A_{2,3}^{(3)} & \cdots & A_{2,mp-1}^{(p-1)} & A_{2,mp}^{(0)} \\ & A_{3,2}^{(3)} & A_{3,3}^{(4)} & \cdots & A_{3,mp-1}^{(0)} & A_{3,mp}^{(1)} \\ & & \ddots & & \vdots & \vdots \\ & & & & A_{mp,mp-1}^{(p-3)} & A_{mp,mp}^{(p-2)} \end{bmatrix},$$

where the superscript indicates the processor assignment.

3.2. Basic parallel algorithm. We start describing the basic parallel QR algorithm by considering the Francis HQR step and finish this subsection with a brief look at the overall algorithm. The basic loop in Figure 1, indexed by i, is generally referred to as "chasing the bulge." Figure 2 gives an arbitrary example of $i = 7$, in a 20×20 matrix, where the integers indicate nonzero elements of the upper Hessenberg matrix, and the B's indicate the fill-in at that stage of the loop. In the next step of the Francis HQR step, a Householder transformation P_i is computed and applied from the left and the right. The affected elements are within the dashed lines. After this application of the Householder transformation, the bulge moves down the diagonal one position, and the next iteration starts.

```
                     +--------+
  0  0  0  0  0  1  1| 1  1  1| 2  2  2  2  2  3  3  3  3  3
  0  0  0  0  0  1  1| 1  1  1| 2  2  2  2  2  3  3  3  3  3
     0  0  0  0  1  1| 1  1  1| 2  2  2  2  2  3  3  3  3  3
        0  0  0  1  1| 1  1  1| 2  2  2  2  2  3  3  3  3  3
           0  0  1  1| 1  1  1| 2  2  2  2  2  3  3  3  3  3
              1  2  2| 2  2  2| 3  3  3  3  3  0  0  0  0  0
                 2  2| 2  2  2| 3  3  3  3  3  0  0  0  0  0
                  +--+--------+-----------------------------+
                  | 2| 2  2  2| 3  3  3  3  3  0  0  0  0  0|
                  | B| 2  2  2| 3  3  3  3  3  0  0  0  0  0|
                  | B| B  2  2| 3  3  3  3  3  0  0  0  0  0|
                  +--+--------+-----------------------------+
                     |       3| 0  0  0  0  0  1  1  1  1  1
                     +--------+
                                0  0  0  0  0  1  1  1  1  1
                                   0  0  0  0  1  1  1  1  1
                                      0  0  0  1  1  1  1  1
                                         0  0  1  1  1  1  1
                                            1  2  2  2  2  2
                                               2  2  2  2  2
                                                  2  2  2  2
                                                     2  2  2
                                                        2  2
```

FIG. 2. *Data decomposition and chasing of the bulge.*

Figure 2 also demonstrates the possible parallelism in the Francis HQR step: the application of P_i to the appropriate elements of H, referred to as a reflection, is perfectly distributed among the processors, except for the processor that owns the bulge and the fill-in elements. This processor must do slightly more work (associated with the intersection of the two boxes). *It is the parallelism within the application of each Householder transformation that is the key to the success of the approach.*

The entire Francis HQR step now parallelizes as follows:

1. The processor(s) that holds $\operatorname{eig}(H(n-1:n, n-1:n))$ computes the shifts and sends them to the first processor. Alternatively, all processors could compute the shifts, since it is a lower-order operation.
2. The processor that owns the data required to compute P_0 computes it and then broadcasts it to all other processors.
3. All processors update their portions of the matrix according to P_0.
4. **for** $i = 1, \ldots, n-2$,
   (a) the processor that owns the data required to compute P_i computes it and then broadcasts it to all other processors;
   (b) all processors update their portions of the matrix according to P_i.

Naturally, every time the computation hits data that lies on the boundaries between processors, a limited amount of communication is required to bring appropriate rows and columns together.

The overall Schur decomposition involves more than just the application of Francis HQR steps. Deflation decreases the problem size as subdiagonals converge to zero. Since a Francis HQR step requires an unreduced Hessenberg matrix, at some point in the Schur decomposition we have the following matrix [18]:

$$H = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ & H_{22} & H_{23} \\ & & H_{33} \end{bmatrix},$$

where H_{33} has already converged and H_{22} is the largest unreduced Hessenberg matrix above H_{33} in the current iteration. One might worry that the property of each row and column being

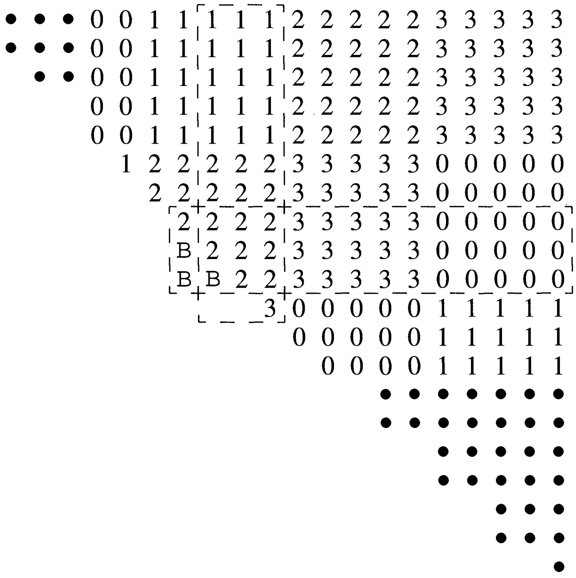


FIG. 3. *Chasing of the bulge after deflation.*

distributed amongst all the processors no longer applies to H_{22}. In the Schur decomposition, however, H_{12} and H_{23} are also involved in the computation (see Figure 3) as well as the Schur vectors, so that the overall work distribution remains equal even after deflation has reduced the problem size. Again, we point out that it is the parallelism within the application of each Householder transformation that is the key to the success of this approach. In §4, experimental results confirm that the overall performance of the code is fairly accurately modeled by the first few iterations.

3.3. Analysis of the simple parallel algorithm. Before discussing the finer details of the parallel code, let us analyze the simple algorithm given above. This will give us an idea of how well the basic approach works. For simplicity, we restrict our timing analysis to a single Francis step. We justified this restriction in the previous subsection and will further justify it with discussion of experimental results in §4. We also ignore shift derivation, convergence, or decoupling set-up costs. We show later how this extra work can be done without incurring excessive overhead.

3.3.1. Cost analysis. For our analysis, we will use the following model: performing a floating point computation requires time γ. Sending a message of length n between two nodes requires time $\alpha + n\beta$, where α represents the latency, and β the penalty per double precision number transferred. For convenience, we shall assume all communication is to the nearest neighbor (processor in a network), and hence no network conflicts occur. Processors can send and receive simultaneously.

We start by computing the sequential expense of the algorithm: computing a Householder transformation from a vector of length three requires C, which are some small constant flops. Applying such a Householder transformation of size three to three rows or columns of length n requires $10n$ flops. Within the loop indexed by i, each transformation is applied to $n - i + 1$ triplets of rows and $i + 3$ triplets of columns. Applying the transformations to the Hessenberg matrix therefore requires $10(n + 4)$ flops and applying the transformations to the Schur matrix requires $10n$ flops. The total cost of executing a single Francis step thus becomes approximately

$$\sum_{i=1}^{n-1}(C + 10(n+4) + 10n)\,\gamma = 20n^2\gamma + O(n).$$

Now turning to the parallel implementation, contributions to the critical path are given roughly as follows: computing a transformation still contributes C flops. Broadcasting this transformation requires time $\alpha + 3\beta$, since the parallel computation can be arranged to pipeline around the logical ring of processors.[1] After this, the processor that computed the transformation must update its part of the rows and columns. The row and column vectors are of length approximately n/p, with a few extra introduced by the bulge. This contributes the equivalent of about $20(n/p+4)$ flops. Now, the processor that computed the current transformation can start computing the next transformation, and the process repeats itself $n-1$ times. The total contribution becomes

$$\sum_{i=1}^{n-1}\left(C\gamma + 10\left(\frac{n}{p}+4\right) + 10\frac{n}{p}\gamma + \alpha + 3\beta\right) = Cn\gamma + 20\frac{n^2}{p}\gamma + 40n\gamma + n\alpha + 3n\beta + O(1). \tag{1}$$

Here we cannot ignore $O(n)$ terms, since if p is comparable to n, these terms matter.

Equation (1) ignores the communication required when a boundary of processors is reached, which happens every h iterations. Each "border" row contains roughly $(n-i)/p$ elements per processor, and each "border" column contains roughly i/p elements per processor. Using transformations of size three implies that when the boundary of processors is reached, the next transformation will also cross the boundary. Hence $2n/p$ items must be communicated to the right and returned. This means that the total communication volume for "border" or boundary data is

$$\frac{n}{h}\left(4\frac{n}{p}\right). \tag{2}$$

The communication volume in (2) can be done in anywhere from one to eight messages depending on the implementation. Eight messages are required when both rows and both columns for the two iterations are sent separately and returned again. One message is possible if data are only sent rightward around the ring, and the results accumulated into a buffer and sent back in $O(1)$ communications at the end of the Francis step. For the purpose of our model, we assume two communications, requiring time

$$2\frac{n}{h}\left(\alpha + 2\frac{n}{p}\beta\right) \tag{3}$$

for a total estimated parallel time of

$$T_{\text{est}}(n,p,h) = Cn\gamma + 20\frac{n^2}{p}\gamma + 40n\gamma + n\alpha + 3n\beta + 2\frac{n}{h}\alpha + 4\frac{n^2}{hp}\beta + O(1). \tag{4}$$

We conclude with a few additional comments.

[1] That is, because of the pipeline between computation and communication, the effective overhead is the cost for the first message transfer to the nearest neighbor. While the second transfer of the same message starts on the next node, the first node has already started the computation.

- Equation (4) ignores a slight disruption in the pipeline of communication and computation, because it is no longer the processor that computed the current transformation that also computes the next transformation. This disruption is minor and we will ignore it.
- It may appear that at the end of a single Francis step, the processors must synchronize. However, if P_{p-2} is the last processor to compute a transformation, P_0 will complete its computation early in the pipeline and will have enough information to start the next Francis step before the pipeline is flushed.[2] If P_{p-2} is not the last processor to compute a transformation, a slight bubble in the pipeline can be expected. This could occur depending on the exact mapping of data used or after deflation.

However, (4) does reflect the expected cost per iteration during the first few iterations. Since experimental results show that the first few iterations quite accurately predict the overall performance of an implementation that includes deflation, we will use (4) for subsequent analysis, ignoring the stated deficiencies in our model.

3.3.2. Scalability. On current generation architectures, the above model shows that the bulk of inefficiency comes from the *number* of messages generated, since α is typically several orders of magnitude greater than either β or γ. To investigate the scalability of the approach, we start by computing the estimated speedup, based on the estimated cost of a single Francis step:

$$\text{(5)} \quad \text{speedup}_{\text{est}}(n, p, h) = \frac{T_{\text{seq}}(h)}{T_{\text{est}}(n, p, h)} = \frac{20n^2}{20\frac{n^2}{p}\gamma + 4\frac{n^2}{hp} + O(n)} = \frac{p}{1 + \frac{4}{h} + O(\frac{p}{n})}.$$

If h is chosen big enough, e.g., $h \approx n/p$, the second term in the denominator also becomes $O(p/n)$, and we can make the following observations: if p is fixed, and n is allowed to grow, eventually perfect speedup will be approached. Furthermore, the estimated efficiency attained is given by

$$\text{(6)} \quad \text{efficiency}_{\text{est}}(n, p, h) = \text{speedup}_{\text{est}}(n, p, h) = \frac{1}{1 + O(\frac{p}{n})}.$$

From this last equation, we can say something about the scalability of the implementation: in order to maintain efficiency, we must grow n linearly with p. This poses a serious problem: memory grows linearly with p, but memory requirements grow with n^2. We conclude that this approach to parallelization will not scale, since memory constraints dictate that eventually efficiency cannot be maintained, even if the problem is allowed to grow to fill the combined memories.

The previous modeling extends to the overall algorithm in a simple manner. On average, to find the eigenvalues of a matrix of size n requires $2n$ HQR iterations. In every other iteration, there is typically a deflation of size one or two. Since deflation occurs less frequently in the beginning than the end, our overall model satisfies

$$\text{overall flops } = \sum_{k=1}^{n} 3 * 20k^2 = 20n^3,$$

where 3 is a heuristic fudge factor. Compare, for example, [18] which uses a similar heuristic, but different fudge factor, to suggest overall flops at $25n^3$.

Similarly, we can extend all of the formulas in the previous subsection. In (4), the number of communication start-ups is $n + 2n/h$. For the overall algorithm, before the bundling

[2] Recall that every other processor owns diagonal blocks in an antidiagonal mapping.

described in §3.4.4, we can estimate

$$\text{communication start-ups } = \sum_{k=1}^{n} 3 * \left(k + \frac{2k}{h}\right) = \left(\frac{3}{2} + \frac{3}{h}\right) n^2.$$

3.3.3. Theoretical limitations. For many dense linear algebra algorithms, better scalability can be obtained by using a so-called two-dimensional data decomposition. Examples include the standard decomposition algorithms, like LU, QR, and Cholesky factorization [13, 19], as well as the reduction to condensed form required for reducing a general matrix to upper Hessenberg form [6, 12]. It is therefore natural to try to reformulate the parallel QR algorithm in an attempt to extract an algorithm that has better scalability properties. In this section we show *theoretically* that there is an *inherent* limitation to the scalability of the QR algorithm that indicates that a more complicated data and work decomposition alone *cannot* improve the scalability of the parallel implementation.

We present a simple analysis of the theoretical limitations of the QR algorithm. There is a clear dependence in the chasing of the bulge that requires at least $O(n)$ steps to complete one Francis HQR step. The associated computation can be thought of as running down the diagonal of the matrix. Moreover, the last element of the matrix must be computed before the next Francis step can start, since it is needed to compute the shift. Since there are $O(n^2)$ computations performed in one such step, the speedup is limited to $O(n)$, which indicates that the maximal number of processors that can be usefully employed is $O(n)$. Hence, $p \leq Cn$ for some constant C.

Let us now analyze the implications of this: if efficient use of the processors is to be attained, the equation $p \leq Cn$ must hold for some constant C. The total memory requirements for a problem of size n is Kn^2 for some constant K. Hence,

$$\text{memory requirements} = Kn^2 \geq \frac{K}{C^2}p^2 = O(p^2).$$

Notice, however, that memory only grows linearly with the number of processors. We conclude that due to memory restrictions, it is impossible to solve a problem large enough to maintain efficiency as processors are added. In the conclusion, we indicate how it may be possible to overcome this apparent negative result.

3.4. Refinements of the parallel algorithm. The above analysis shows that there is reason to believe that trying to generalize the Hankel-wrapped mapping to yield the equivalent of a two-dimensional wrapping will not result in the same kinds of benefits as it would for a factorization algorithm. However, there are a number of refinements that can be made that will improve the performance of the parallel QR algorithm. Moreover, the algorithm as it was described in §3.2 is far from a complete implementation.

3.4.1. Accumulating Q. The algorithm in §3.2 ignores the accumulation of the orthogonal matrix Q. There is a perfect parallelism in this operation, requiring no more communication than must already be performed as part of the updating of H.

Each stage of the Francis HQR step in Figure 1 requires the computation $Q \leftarrow QP_i$. Since P_i is broadcast during the update of the Hessenberg matrix, all processors already have this information. The computation $Q \leftarrow QP_i$ reflects three columns of Q. Which three get reflected depends on i. Since Q is not accessed elsewhere within the HQR kernel, we can assume Q has the best data storage possible for reflecting three columns. This data storage is a one-dimensional row mapping. With a one-dimensional row mapping, the data in any row is always contained on a single processor. Each row in a column transform can be computed independently. So there is no extra communication required to complete the column transform, and this extra work is embarrassingly parallel.

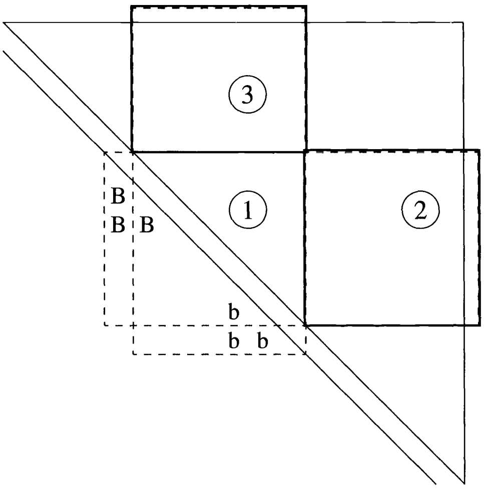


FIG. 4. *Bundling Householder transformations and order of updates.*

3.4.2. Deflation. We previously mentioned that as subdiagonal elements become numerically zero, the process proceeds with unreduced submatrices on the diagonal. This process is known as deflation. One convenient way to determine the points where such a block start and end is to broadcast enough diagonal information during the bulge chase so that at the end of a complete HQR step, all processors hold all information required to determine when breakpoints occur. After rotating columns i, $i+1$, and $i+2$, column i remains fixed for the remainder of that Francis HQR step. Therefore, the three elements of the tridiagonal portion of H in column i ($H(i-1:i+1, i)$) can also be broadcast with the three elements defining the Householder reflection at step i. This requires sending six elements instead of three per transformation. While this creates a certain redundancy, the overhead is $O(n)$, and hence lower order.

3.4.3. Determining shifts. Related to the refinement mentioned above concerning deflation, it is convenient to broadcast all entries of the tridiagonal of the newly formed Hessenberg matrix, as we describe in §3.4.2, since this allows all processors access to the data required to compute the next shifts.

3.4.4. Bundling transformations. As is argued in §3.3, the bulk of the communication overhead is due to communication start-up cost. This cost can be amortized over several computations and applications of Householder transformations by bundling several Householder transformations before broadcasting.

We direct the reader's attention to Figure 4.

This picture represents the data in the block that currently holds the bulge. This block is assumed to be completely assigned to one processor. In chasing the bulge a few positions, to where it moves from the position indicated by "B"s to the position indicated by "b"s, a number of Householder transformations are computed. To do so, only the computation associated with the region marked by "1" needs to be performed.[3] Thus these transformations can be bundled by performing this minimal computation, followed by the broadcast, after which the other regions, marked by "2" and "3" can be updated. The net effect is that the number of start-ups

[3] As is noted in Henry [20, 23], the minimal information needed to determine the upcoming transforms is roughly eighty percent of the total flops required to actually complete the entire update in region "1" of Figure 4. This "partial" step is referred to as a lookahead step and can also be used to determine better shifts (cf. Henry [22]).

is reduced by the number of transformations that are bundled. Naturally, bundling across processor boundaries becomes complicated and has only marginal benefit.

Repeating the analysis in §3.3 above, we get the following estimates: computing r transformations, updating only region "1" in Figure 4 requires approximately

$$r(C + 10(r + 4))\gamma$$

time. Broadcasting the transformations requires time

$$\alpha + 3r\beta.$$

Next, updating regions "2" and "3" on the processor that computes the transformations requires time

$$10r\left(\frac{n}{p} - r\right)\gamma$$

to update matrix A, and

$$10r\frac{n}{p}\gamma$$

to update matrix Q. This is repeated n/r times, and border information must be communicated every h transformations, giving a (very rough) total estimated time of

$$20\frac{n^2}{p}\gamma + Cn\gamma + 40n\gamma + \frac{n}{r}\alpha + 3n\beta + 2\frac{n}{h}\alpha + 4\frac{n^2}{hp}\beta + O(1).$$

Clearly the above estimate is only a rough estimate: it suggests that increasing bundling factor r arbitrarily will continue to improve total time. Notice that this estimate is only reasonable when the pipe is undisturbed. However, every time a boundary is encountered, the boundary will be disturbed and therefore larger bundling factors will increase load imbalance. Nonetheless, our model gives us insight.

4. Performance results. In this section, we report the performance of a prototype working implementation of the described parallel implementation. Our implementation is a parallelization of the LAPACK [1] routine DLAHQR, and is mathematically exact except that the sequential routine applies one final rotation per converged eigenpair so that the resulting "Schur" form has some regularity. Hence, there is no need to report in detail on the numerical accuracy of the implementation: all numerical properties of the LAPACK code apply to this parallel implementation.

We report performance obtained on an Intel Paragon™ XP/S Model 140 Supercomputer, running SUNMOS version S1.4.8. The following is crucial to our analysis:

> Since finding all eigenvalues of a very large matrix takes a considerable amount of time (measured in days) we report the performance of the algorithm during the first five iterations. For problems of size 2750 or less, and for 64 processors or less, we saw extrapolating total time from measuring five iterations always yielded a predicted performance that was within three percent measured performance when the code was executed to completion.[4]

In addition, it is hard to measure speedup with respect to a single processor, due to insufficient memory to hold a large matrix. As a result, we measured the performance of the best-known sequential QR algorithm and report the parallel performance with respect to that: the single

[4] This is possible because we restricted our timings to a Schur decomposition.

TABLE 1
Parallel HQR.

p	n	Scaled speedup	Efficiency
4	2000	2.5	.63
8	2800	5.2	.65
16	4000	10.0	.63
32	4800	17.1	.53
48	6000	23.0	.48
64	8000	30.1	.47
96	9600	37.4	.39

node performance of the fastest QR algorithm (not necessarily a standard double shift algorithm from LAPACK) peaks at 10 Mflops for the five iterations. We report scaled speedup as being the speedup obtained with respect to the calculated time required to run a given problem on a single node performing at 10 Mflops. In this case, actual flops within the algorithm were calculated and used for results, and not heuristics.

Many parameters are used to describe the parallel implementation, including the number of nodes used p, the matrix size n, the blocking factor h, and the bundling factor r. All the runs in Table 1 were for problems of size approximately 10 to 14 Mbytes per processor, all with blocking factors h given by n/p, with a bundling factor of $r = 2$. Our models predict that varying h and r impacts communication overhead incurred. However, in practice , the major source of inefficiency appears to come from the fact that row and column updates do not have the same memory access patterns and therefore execute at different rates. Our experiments indicate row transforms are 13 percent faster than column transforms. At each step there is a roughly even amount of work for all the nodes, but there is not the same amount of row or column work. This means that at *each* step there is a delay for a few of the nodes to complete the ring broadcast. Hence, although our model predicts the load balance to be quite good, the different rates of execution make it much less well balanced. This shows up in the timings given in Table 1. Notice that, at best, about 65 percent efficiency is attained.

5. Conclusion. The research described in this paper was meant as a response to claims that no parallelism exists in the QR algorithm. The theoretical results in this paper show there are approaches to implementing the QR algorithm that allow parallelism to be extracted in a natural way. The actual implementation of this method on a high performance parallel computer shows that, in practice, parallelism can be extracted as well.

We caution the reader against misinterpreting the results in this paper as largely supporting the notion that new methods like those developed by Bai and Demmel [4] and Dongarra and Sidani [11] must be pursued if nonsymmetric eigenvalue problems are to be solved on massively parallel computers. Let us address some of the arguments that can be made to support such an interpretation and how these arguments are somewhat unsatisfactory.

No parallelism exists in the nonsymmetric QR algorithm. We believe the major results in this paper are a counterexample to this statement, both in theory and practice.

The performance of even the sequential algorithm leaves something to be desired. One argument for matrix–matrix multiplication-based methods is that matrix multiplication can achieve much higher performance rates than a kernel that applies a Householder transformation. This more than offsets the added floating point operations required for such novel solutions. This argument is true only because the matrix–matrix multiplication is recognized as an important kernel that warrants highly optimized implementation. It is the data reuse available in the matrix–matrix multiplication that allows near-peak performance to be achieved on a large number of platforms. However, a careful analysis of reuse of data when *multiple*

Householder transformations are applied, which could be exploited with bundling, shows that the performance of the sequential QR algorithm could be greatly improved if such a kernel were assembly coded. Clearly a user who can afford a massively parallel computer would be able to afford to assembly code such a kernel.

One could argue that assembly coding this kernel would only improve the 40 percent of total time spent in useful computation and hence would actually *decrease* efficiency. However, we argue that since most overhead is due to load imbalance, the total time should benefit, not just the time spent in useful computation.

Parallel QR algorithms cannot be scalable. One could draw this conclusion from §3.3.3. However, notice that the analysis holds for the algorithm when only a single double-shift is used during each iteration. If a method is parallelized that uses s shifts, the total computation per iteration is increased to $O(sn^2)$, while the dependence during the bulge chase only increases to $O(n+s)$ if multiple bulges are chased in a pipelined fashion. We conclude that the limit on speedup is now given by $O(sn^2)/O(n+s) \approx O(sn)$. $O(sn)$ processors can be usefully employed, leading to memory requirements

$$\text{memory requirements} = Kn^2 >= O\left(\frac{p^2}{s^2}\right).$$

As long as $p/s^2 = O(1)$, i.e., $s = O(\sqrt{p})$, it may be possible to achieve scalability while meeting the fixed memory per processor criteria.

Scalable implementations of the QR algorithm may be achievable for parallel implementations that use s shifts simultaneously, provided the number of shifts grows with the square-root of the number of processors.

We must note that our data decomposition inherently requires n to grow with p and therefore it will not provide for the required parallel implementation. Therefore, generalizations to two-dimensional data decompositions will be necessary.

Notice that some of the above comments indicate that many of the observations made from the earlier sections were merely due to the fact that in this paper we restricted ourselves to discussing the parallel implementation of widely used QR algorithms.

In contrast, we mention the advantages of using the QR algorithm:

- Fewer flops, fast convergence. In many cases (cf. [5]), other algorithms can require several times the number of floating point calculations (although they can be in matrix multiplication).
- High accuracy. The QR algorithm has always remained one of the most accurate algorithms for the unsymmetric Schur decomposition.
- Parallelism. This paper demonstrates that for a moderate-sized system, parallelism is possible.

We conclude by saying that the reason parallel QR algorithm implementations may not be competitive is because insufficient resources have been allocated to study their implementation. Those resources are currently being used instead to explore novel algorithms. If someone has the required resources, our research points clearly to what properties a parallel implementation must have to be successful.

Acknowledgments. Our final runs were made on Sandia's Paragon™ XP/S Model 140 Supercomputer running SUNMOS S1.4.8, and we are grateful for this resource and operating system. SUNMOS stands for Sandia and University of New Mexico Operating System and is a joint collaboration between Sandia National Laboratories and UNM. SUNMOS is copyrighted by Sandia National Laboratories. Most of the development was done on various Paragon™ supercomputers at Intel Supercomputer Systems Division. We thank Ken Stanley and David Watkins for communications regarding this work.

REFERENCES

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorenson, *LAPACK Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.

[2] L. Auslander and A. Tsao, *On parallelizable eigensolvers*, Advanced Appl. Math., 13 (1992), pp. 253–261.

[3] Z. Bai and J. Demmel, *On a block implementation of Hessenberg multishift QR iteration*, Internat. J. High Speed Computing, 1 (1989), pp. 97–112.

[4] ———, *Design of a parallel nonsymmetric eigenroutine toolbox, part* I, in Parallel Processing for Scientific Computing, R. Sincovec, D. Keyes, M. Leuze, L. Petzold, and D. Reed, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1993, pp. 391–398.

[5] Z. Bai, J. Demmel, J. Dongarra, A. Petitet, H. Robinson, and K. Stanley, *The Spectral Decomposition of Nonsymmetric Matrices on Distributed Memory Parallel Computers*, LAPACK Working Note 91, University of Tennessee, Knoxville, Jan. 1995.

[6] M. W. Berry, J. J. Dongarra, and Y. Kim, *A Highly Parallel Algorithm for the Reduction of a Nonsymmetric Matrix to Block Upper-Hessenberg Form*, LAPACK Working Note 68, CS-94-221, University of Tennessee, Knoxville, Feb. 1994.

[7] R. Byers, *Numerical stability and instability in matrix sign function based algorithms*, in Computational and Combinatorial Methods in Systems Theory, C. Byrnes and A. Lindquist, eds., North–Holland, Amsterdam, 1986, pp. 185–200.

[8] D. Boley, *Solving the generalized eigenvalue problem on a synchronous linear processor array*, Parallel Comput., 3 (1986), pp. 123–166.

[9] D. Boley and R. Maier, *A Parallel QR Algorithm for the Nonsymmetric Eigenvalue Problem*, Technical Report TR-88-12, Dept. of Computer Science, University of Minneapolis, Minneapolis, 1988.

[10] J. W. Demmel, *LAPACK Working Note* 47: *Open Problems in Numerical Linear Algebra*, Technical Report CS-92-164, University of Tennessee, May 1992.

[11] J. J. Dongarra and M. Sidani, *A parallel algorithm for the nonsymmetric eigenvalue problem*, SIAM J. Sci. Statist. Comput., 14 (1993), pp. 542–569.

[12] J. J. Dongarra and R. A. van de Geijn, *Reduction to condensed form on distributed memory architectures,* Parallel Comput., 18 (1992), pp. 973–982.

[13] J. J. Dongarra, R. A. van de Geijn, and D. Walker, *Scalability issues affecting the design of a dense linear algebra library*, J. Parallel Distributed Comput., 22 (1994), pp. 523–537.

[14] P. J. Eberlein, *On the Schur decomposition of a matrix for parallel computation*, IEEE Trans. Comput., C-36 (1987), pp. 167–174.

[15] J. G. F. Francis, *The QR transformation: A unitary analogue to the LR transformation, Parts* I *and* II, Comp. J., 4 (1961), pp. 332–345.

[16] G. A. Geist and G. J. Davis, *Finding eigenvalues and eigenvectors of unsymmetric matrices using a distributed memory multiprocessor*, Parallel Comput., 13 (1990), pp. 199–209.

[17] G. A. Geist, R. C. Ward, G. J. Davis, and R. E. Funderlic, *Finding eigenvalues and eigenvectors of unsymmetric matrices using a hypercube multiprocessor*, in Proceedings of the Third Conference on Hypercube Concurrent Computers and Applications, Pasadena, CA, G. Fox, ed., 1988, pp. 1577–1582.

[18] G. Golub and C. F. Van Loan, *Matrix Computations*, 2nd ed., The John Hopkins University Press, Baltimore, MD, 1989.

[19] B. A. Hendrickson and D. E. Womble, *The torus-wrap mapping for dense matrix calculations on massively parallel computers.*, SIAM J. Sci. Statist. Comput., 15 (1994), pp. 1201–1226.

[20] G. Henry, *Improving the unsymmetric parallel QR algorithm on vector machines*, in Proc. of the 6th Conference of Parallel Processing, R. Sincovec et al., eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1993, pp. 353–357,

[21] ———, *Increasing Data Reuse in the Unsymmetric QR Algorithm*, Technical Report CTC92TR100, Theory Center, Cornell University, Ithaca, NY, July 1992.

[22] ———, *A New Approach to the Schur Decomposition*, Technical Report, Theory Center, Cornell University, Ithaca, NY, manuscript.

[23] ———, *Improving Data Re-Use in Eigenvalue-Related Computations*, Ph.D. thesis, Cornell University, Ithaca, NY, Jan. 1994.

[24] I. C. F. Ipsen and Y. Saad, *The impact of parallel architectures on the solution of eigenvalue problems*, in Large Scale Eigenvalue Problems, J. Cullum and R. Willoughby, eds., Elsevier–North Holland, Amsterdam, 1986, pp. 37–49.

[25] I. C. F. Ipsen, Y. Saad, and M. H. Schultz, *Complexity of dense-linear-system solution on a multiprocessor ring*, Linear Algebra Appl., 77 (1986), pp. 205–239.

[26] J. W. Juszczak and R. A. van de Geijn, *An experiment in coding portable parallel matrix algorithms*, in Proceedings of the Fourth Conference on Hypercube Concurrent Computers and Applications, Monterey, CA, 1989, pp. 675–680.

[27] S. Lederman, A. Tsao, and T. Turnbull, *A Parallelizable Eigensolver for Real Diagonalizable Matrices with Real Eigenvalues*, Technical Report TR-91-042, Supercomputing Research Center, 1991.

[28] C. B. Moler, *MATLAB User's Guide*, Technical Report CS81-1, Department of Computer Science, University of New Mexico, Albuquerque, 1980.

[29] A. Purkayastha, *A Parallel Algorithm for the Sylvester-Observer Equations*, Ph.D. dissertation, Northern Illinois University, DeKalb, IL, Jan. 1993.

[30] G. W. Stewart, *A parallel implementation of the QR algorithm*, Parallel Computing, 5 (1987), pp. 187–196.

[31] R. A. van de Geijn, *Implementing the QR-Algorithm on an Array of Processors*, Ph.D. thesis, TR-1897, Department of Computer Science, University of Maryland, College Park, MD, 1987.

[32] ———, *Storage schemes for parallel eigenvalue algorithms*, in Numerical Linear Algebra, Digital Signal Processing and Parallel Algorithms, G. Golub and P. Van Dooren, eds., Springer-Verlag, Berlin, New York, 1988, pp. 639–648.

[33] R. A. van de Geijn and D. G. Hudson, *An efficient parallel implementation of the nonsymmetric QR algorithm*, in Proceedings of the 4th Conference on Hypercube Concurrent Computers and Applications, Monterey, CA, 1989, pp. 697–700.

[34] D. Watkins, *Shifting strategies for the parallel QR algorithm*, SIAM J. Sci. Comput., 15 (1994), pp. 953–958.

[35] ———, *Transmission of shifts in the multishift QR algorithm*, Proceedings of the 5th SIAM Conference on Applied Linear Algebra, Snowbird, UT, June 1994, pp. 302–305.

[36] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, Oxford, 1965.

SIAM J. SCI. COMPUT.
Vol. 17, No. 4, pp. 884–905, July 1996

AN OVERDETERMINED SCHWARZ ALTERNATING METHOD*

HUOSHENG SUN† AND WEI-PAI TANG†

Abstract. Developing modern extensions of the Schwarz alternating method (SAM) has been a primary focus in the research of domain decomposition during the past 10 years. Among the various research efforts, the generalized Schwarz alternating method (GSAM) was an attempt to use the Robin condition

$$\omega u + (1 - \omega)\frac{\partial u}{\partial n}$$

on those artificial boundaries to improve the performance of the SAM. Its convergence rate is much faster than the classical SAM (with a large overlap), yet only a minimum overlap is required. Unfortunately, sensitivity of the convergence of the GSAM to the parameter ω has limited its practical applications. In this paper, a new kind of coupling is proposed which possesses the same benefits as the GSAM. The advantage of our new algorithm over the GSAM is that the optimal convergence rate is achieved on a wider range of the parameter. That is, selection of the optimal parameter is not crucial to the new algorithm's performance. Numerical tests have been carried out for a variety of difficult problems including nonsymmetric and indefinite problems.

Key words. Schwarz alternating method, generalized Schwarz splitting, generalized Schwarz alternating method, overdetermined Schwarz alternating method, domain decomposition, parallel computation, overlap

AMS subject classifications. 65F10, 65N10

1. Introduction.

Schwarz-type alternating methods have become some of the most important approaches in domain decomposition techniques during the past few years [2, 3, 5, 7, 9–11, 13, 15–18, 20]. The theoretical foundations for the classical Schwarz alternating method (SAM) and its modern extension are maturing [11, 19, 21]. Our list of references can only reflect a small portion of the studies done in this direction. Current popular approaches are seeking superior preconditioners to improve performance. A multilevel preconditioner, for example, is a typical technique used by many [3, 17, 19, 21]. However, one of the most interesting aspects of the SAM, namely, the coupling on those artificial boundaries, has not received enough attention. Our studies show that further investigation in this direction can be promising as well.

It is well known that the amount of overlap affects the rate of convergence, yet large overlap is not desirable. Tang proposed *generalized Schwarz splitting* a few years ago [18]. One special element of his approach to the solution of an elliptic PDE is to use the Robin boundary condition

$$\omega u + (1 - \omega)\frac{\partial u}{\partial n}$$

on the artificial boundaries. P. Lions [11] presented a very similar idea, "a variant for nonoverlapping subdomains," in using the Robin condition on the interfaces between subregions.

It was shown that a fast convergence can be achieved with minimum overlap if an optimal ω is chosen [18]. The convergence is even faster than with the classical SAM with a large overlap. The minimum overlap requirement is, of course, an attractive feature of the generalized Schwarz alternating method (GSAM). Unfortunately, the analysis also indicates that the convergence can be sensitive to this parameter. In particular, to identify a robust algorithm for estimating the optimal parameter in real applications is an extremely difficult task. An

*Received by the editors December 9, 1993; accepted for publication (in revised form) March 9, 1995. This research was supported by the Natural Sciences and Engineering Research Council of Canada, and by the Information Technology Research Centre, which is funded by the Province of Ontario.

†Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1.

attempt to use a similar idea in a symmetric successive overrelaxation (SSOR) for removing the sensitivity is not successful.

Therefore, retaining the good features of the GSAM, namely fast convergence and the minimum overlap requirement, and eliminating the sensitivity of any parameter to the convergence, are the two most important characteristics we need for the new approaches.

It is easy to see that the motivation behind the GSAM is to use the information of both the unknown and its derivative in the coupling of the neighboring subdomains. A weighted combination of the unknown and its derivative forces a contribution of both. If the combination is proper, a fast convergence with a minimum overlap can be achieved. This combination, therefore, poses a "*stronger*" constraint, or consistency of the neighboring solutions, on those artificial boundaries. On the other hand, the most forceful combination of this constraint can only be reached in a very narrow range of the parameter ω. Though the GSAM suffers from its practical implications, it demonstrates a successful attempt to use a stronger coupling. If a more vigorous coupling between the solutions of those neighboring subdomains can be introduced, the ideal goal may become reality.

Based on the above observation, we are looking for alternative couplings which can be introduced between the neighboring subdomains. In fact, the enforcement of the solutions between the two neighboring subdomains may not necessarily be just on the artificial boundaries. On the whole overlapping part of the two neighboring subdomains, the two solutions also have the same values for both the unknown and its derivative. Therefore, we can introduce an artificial boundary layer for the corresponding artificial boundary. The Robin condition is imposed on the entire layer. The primary motive of the new approach is to impose the constraints on the boundary layer rather than on a single boundary as the GSAM did. We hope that this stronger enforcement will be beneficial to the convergence behavior of the new approach.

In the next section, the new method, the overdetermined Schwarz alternating method (OSAM), is introduced. Section 3 describes the technique for the discretization of the artificial boundary layer. The analysis of the convergence for the model problem is shown in §4. Numerical testing results are shown in §5. The last section concludes.

2. Overdetermined SAM. Let L be an elliptic operator

$$L = -\frac{\partial}{\partial x}\left(w_1\frac{\partial}{\partial x}\right) - \frac{\partial}{\partial y}\left(w_2\frac{\partial}{\partial y}\right) + s\frac{\partial}{\partial x} + t\frac{\partial}{\partial y} + c, \tag{1}$$

where w_1, w_2, s, t, and c are functions of x and y which satisfy[1]

$$w_1(x,y),\quad w_2(x,y) \geq C > 0, \qquad c(x,y) \geq 0.$$

Consider the following boundary value problem:

$$\left\{\begin{array}{rcll} L(u) & = & f, & \Omega, \\ u & = & g, & \partial\Omega, \end{array}\right. \tag{2}$$

where Ω is a bounded region in R^2. For simplicity, we consider only the two-overlapping-subdomain case (see Figure 1). The generalization to an irregular solution domain or multisubdomain is straightforward. The rectangular solution region Ω is partitioned into two overlapping subdomains Ω_1 and Ω_2. Let

$$\begin{array}{ll} \Gamma_{21} = \partial\Omega_2 \cap \Omega_1, & \Gamma_{12} = \partial\Omega_1 \cap \Omega_2, \\ \Gamma_1 = \partial\Omega \cap \partial\Omega_1, & \Gamma_2 = \partial\Omega \cap \partial\Omega_2. \end{array}$$

[1] Our methods can also be applied to nonself-adjoint and indefinite cases (see §5).

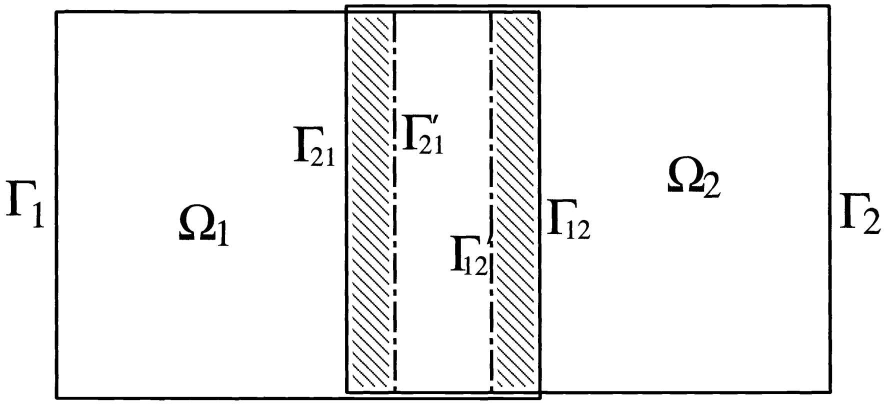


FIG. 1. *Artificial boundary layers of two overlapping subdomains.*

Here, Γ_{12} and Γ_{21} are the artificial boundaries for subdomains Ω_1 and Ω_2, respectively. To make the picture more readable, the Ω_2 has been shifted upwards slightly. We introduce two artificial boundary layers $\mathcal{L}_1$ and $\mathcal{L}_2$ (shaded parts in Figure 1), which are next to the corresponding artificial boundaries Γ_{12} and Γ_{21} (they will be called boundary layers, for simplicity, later). The thickness of these layers depends on the grid. When a uniform grid is employed, the corresponding thickness of the boundary layer will be the grid size h. For a general triangular mesh, the boundary layer will be the union of all triangles for which at least one of its edges or nodes is on the artificial boundary. The motivation for this choice is to allow the minimum overlap needed in the new algorithm. Denote Γ'_{12} and Γ'_{21} as the inner boundaries of the boundary layers $\mathcal{L}_1$ and $\mathcal{L}_2$, respectively.

The Robin condition is imposed on the entire boundary layer. Then, a new problem can be stated as follows:

$$(3)\qquad \begin{cases} L(u_1) = f, & x \in \Omega_1, \\ u_1|_{\Gamma_1} = g, & \\ b(u_1)|_{\mathcal{L}_1} = b(u_2)|_{\mathcal{L}_1}, & \end{cases}$$

$$(4)\qquad \begin{cases} L(u_2) = f, & x \in \Omega_2, \\ u_2|_{\Gamma_2} = g, & \\ b(u_2)|_{\mathcal{L}_2} = b(u_1)|_{\mathcal{L}_2}, & \end{cases}$$

where $b(u) = \omega u + (1-\omega)\frac{\partial u}{\partial n}$. There are infinitely many choices for the function ω. To focus our discussion, we only investigate a very special case in this paper:

$$(5)\qquad \omega = \omega(x, y) = \begin{cases} 1, & (x, y) \text{ is on the artificial boundary,} \\ 0, & \text{otherwise.} \end{cases}$$

Therefore, the constraint on the artificial boundaries is Dirichlet type and Neumann type in the rest part of the boundary layer. Compared to the original SAM, the new form has a stronger coupling between the two solutions of the neighboring subdomains. Problem (3)–(4) is obviously overdetermined. In general, the solution for an overdetermined problem may not exist, and a least square type of solution should be sought. In this case, the solution of (2) is a solution of (3)–(4).

An iterative algorithm for solving this problem can be easily extended from the classical SAM:

```
Choose  $u_2^0|_{\Gamma_{12}}$,  $\frac{\partial u_2^0}{\partial n}\Big|_{\Gamma'_{12}}$
for     $k = 1, 2, \ldots,$
    solve
        $L(u_1^k) = f,$                              $(x, y) \in \Omega_1,$
        $u_1^k|_{\Gamma_1} = g|_{\Gamma_1},$
        $b(u_1^k)|_{\mathcal{L}_1} = b(u_2^{k-1})|_{\mathcal{L}_1},$
    solve
        $L(u_2^k) = f,$                              $(x, y) \in \Omega_2,$
        $u_2^k|_{\Gamma_2} = g|_{\Gamma_2},$
        $b(u_2^k)|_{\mathcal{L}_2} = b(u_1^k)|_{\mathcal{L}_2},$
end  for
```

The implementation and analysis of this algorithm will be shown next.

3. Discretization of the boundary layers. Each of the problems (3)–(4) is overdetermined. To design a deterministic algorithm, reconstruction of the coupling on the boundary layer is necessary. When the problem is discretized, the constraint on the artificial boundary layer affects only the grid nodes on the boundary of the layer (assume the boundary layer is one grid thick), for example, the grid nodes on Γ_{12} and Γ'_{12} of the subdomain Ω_1. The grid nodes of the boundary layer are called inner boundary layer nodes if they are inside the subdomain and outer boundary layer nodes otherwise.

We observe that the solution on Γ'_{12} and Γ'_{21} satisfies both the difference equation[2] and the derivative coupling between two solutions on the neighboring subdomain. Therefore, a natural approach is to apply a weighted combination of these two conditions. It is enough to demonstrate how the two constraints on Γ'_{12} are combined into a single equation. The following part of this section gives the procedure.

Let u^1 and u^2 be the solutions on subdomains Ω_1 and Ω_2, respectively. For each interior node (i, j) in Ω_1, a difference equation

$$Du^1_{i,j} = a^{i,j}_{0,-1}u^1_{i,j-1} + a^{i,j}_{-1,0}u^1_{i-1,j} + a^{i,j}_{0,0}u^1_{i,j} + a^{i,j}_{1,0}u^1_{i+1,j} + a^{i,j}_{0,1}u^1_{i,j+1} = b^{i,j} \tag{6}$$

can be derived using any common discretization strategy.[3] We use $u^1_{i,j}$ to denote the solution u^1 at node (i, j). Let (i, j) be a node next to Γ'_{12} in the subdomain Ω_1. Both this node and its neighboring nodes are indicated by "○". Then, the neighbor node $(i, j+1)$ is located on Γ'_{12}. Both the node $(i, j+1)$ and its neighboring nodes are indicated by "▽" (see Figure 2). By the construction of our boundary layer, it is clear that node $(i, j+2)$ is located on the artificial boundary Γ_{12}. The difference equation at node $(i, j+1)$ is then

$$\begin{aligned} Du^1_{i,j+1} &= a^{i,j+1}_{0,-1}u^1_{i,j} + a^{i,j+1}_{-1,0}u^1_{i-1,j+1} + a^{i,j+1}_{0,0}u^1_{i,j+1} + a^{i,j+1}_{1,0}u^1_{i+1,j+1} + a^{i,j+1}_{0,1}u^1_{i,j+2} \\ &= b^{i,j+1}. \end{aligned} \tag{7}$$

By the Dirichlet condition on Γ_{12}, we have

$$u^1_{i,j+2} = u^2_{i,j+2}. \tag{8}$$

[2] We assume that this boundary value problem has been discretized by finite difference or finite element methods.

[3] Both central and upwind schemes are tested for our new algorithms. Their convergence characteristics are rather similar.

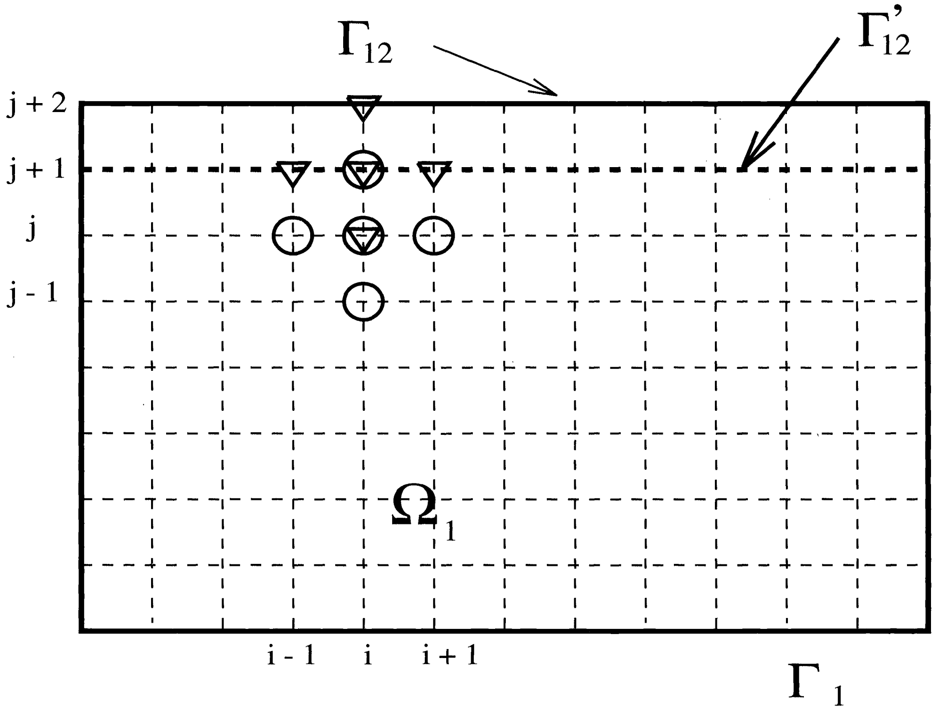


FIG. 2.

Substituting (8) into (7) and solving for $u^1_{i,j+1}$, we get

$$(9)\quad a^{i,j+1}_{0,0}u^1_{i,j+1} = b^{i,j+1} - a^{i,j+1}_{0,-1}u^1_{i,j} - a^{i,j+1}_{-1,0}u^1_{i-1,j+1} - a^{i,j+1}_{1,0}u^1_{i+1,j+1} - a^{i,j+1}_{0,1}u^2_{i,j+2}.$$

If $a^{i,j+1}_{0,0} \neq 0$ (in general, this is true for an elliptic equation), we solve for $u^1_{i,j+1}$ from (9) to get

$$(10)\quad u^1_{i,j+1} = (b^{i,j+1} - a^{i,j+1}_{0,-1}u^1_{i,j} - a^{i,j+1}_{-1,0}u^1_{i-1,j+1} - a^{i,j+1}_{1,0}u^1_{i+1,j+1} - a^{i+1,j}_{0,1}u^2_{i,j+2})/a^{i,j+1}_{0,0}.$$

From the Neumann condition on Γ'_{12}, we have

$$\left.\frac{\partial u^1}{\partial x}\right|_{\Gamma'_{12}} = \left.\frac{\partial u^2}{\partial x}\right|_{\Gamma'_{12}}.$$

In discrete form, at node $(i, j+1)$, this reads

$$(11)\qquad u^1_{i,j+1} = u^1_{i,j} + u^2_{i,j+1} - u^2_{i,j}.$$

Similarly we have, at nodes $(i \pm 1, j+1)$,

$$(12)\qquad u^1_{i\pm1,j+1} = u^1_{i\pm1,j} + u^2_{i\pm1,j+1} - u^2_{i\pm1,j}.$$

Substituting (12) into (10), $u^1_{i,j+1}$ can then be expressed as

$$\begin{aligned} u^1_{i,j+1} = \Big\{ & b^{i,j+1} - a^{i,j+1}_{0,-1}u^1_{i,j} - a^{i,j+1}_{-1,0}(u^1_{i-1,j} + u^2_{i-1,j+1} - u^2_{i-1,j}) \\ (13)\qquad & -a^{i,j+1}_{1,0}(u^1_{i+1,j} + u^2_{i+1,j+1} - u^2_{i+1,j}) - a^{i+1,j}_{0,1}u^2_{i,j+2}\Big\} / a^{i,j+1}_{0,0}. \end{aligned}$$

There are two conditions (13) and (11) for the unknown $u_1^{i,j+1}$ in (6). Let $0 < \theta \leq 1$ be the weight parameter. Then a weighted combination of (13) and (11) gives

$$\begin{aligned} u^1_{i,j+1} = \theta \Big\{ & b^{i,j+1} - a^{i,j+1}_{0,-1} u^1_{i,j} - a^{i,j+1}_{-1,0} (u^1_{i-1,j} + u^2_{i-1,j+1} - u^2_{i-1,j}) \\ & -a^{i,j+1}_{1,0} (u^1_{i+1,j} + u^2_{i+1,j+1} - u^2_{i+1,j}) - a^{i+1,j}_{0,1} u^2_{i,j+2} \Big\} / a^{i,j+1}_{0,0} \\ & +(1-\theta)(u^1_{i,j} + u^2_{i,j+1} - u^2_{i,j}). \end{aligned} \tag{14}$$

Substituting (14) into (6) for any interior nodes next to Γ'_{12}, we can essentially eliminate all the nodes on Γ'_{12} in the final matrix equation for Ω_1.

It should also be pointed out that our new algorithm may appear to require more overlap for the boundary layer. However, after a combination of the two conditions on the boundary layer, only the minimum overlap is really needed for this new approach. In other words, the unknowns on the boundary layer can be eliminated totally from the matrix equation of the subdomain before the iteration starts.

4. Convergence analysis. As shown in [18], Schwarz splitting is a useful tool for analyzing the convergence behavior of the Schwarz-type algorithms. Here, a three-subdomain case is presented. The generalization to more subdomains in the strip case is straightforward.

Consider problem (2), and assume its solution domain is decomposed in the y-direction into three overlapped subdomains. The discretized matrix form can be written as

$$Ax = \begin{pmatrix} A_{11} & A_{12} & 0 & 0 & 0 \\ A_{21} & A_{22} & A_{23} & 0 & 0 \\ 0 & A_{32} & A_{33} & A_{34} & 0 \\ 0 & 0 & A_{43} & A_{44} & A_{45} \\ 0 & 0 & 0 & A_{54} & A_{55} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{pmatrix} = b. \tag{15}$$

The order of the unknowns is arranged so that $\{x_1, x_2\}$ are located in Ω_1, $\{x_2, x_3, x_4\}$ are located in Ω_2, and $\{x_4, x_5\}$ are located in Ω_3. $\{x_2\}$, $\{x_4\}$ correspond to the unknowns in the overlapping parts Ω_{12} and Ω_{23}, respectively.

The GSAM procedure for (2) is equivalent to a 3 by 3 block Gauss–Seidel iteration for the enhanced matrix problem of (15)[18],

$$\tilde{A}\tilde{x} = \begin{pmatrix} A_{11} & A_{12} & 0 & 0 & 0 & 0 & 0 \\ A_{21} & B_2 & C_2 & A_{23} & 0 & 0 & 0 \\ A_{21} & C'_2 & B'_2 & A_{23} & 0 & 0 & 0 \\ 0 & 0 & A_{32} & A_{33} & A_{34} & 0 & 0 \\ 0 & 0 & 0 & A_{43} & B_4 & C_4 & A_{45} \\ 0 & 0 & 0 & A_{43} & C'_4 & B'_4 & A_{45} \\ 0 & 0 & 0 & 0 & 0 & A_{54} & A_{55} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x'_2 \\ x_3 \\ x_4 \\ x'_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_2 \\ b_3 \\ b_4 \\ b_4 \\ b_5 \end{pmatrix} = \tilde{b}, \tag{16}$$

where

$$A_{22} = B_2 + C_2 = C'_2 + B'_2, \quad A_{44} = B_4 + C_4 = C'_4 + B'_4. \tag{17}$$

The splitting in (17) corresponds to the coupling on the artificial boundaries of the GSAM. The corresponding matrix splitting is called a generalized Schwarz splitting (GSS). The analysis of the convergence is, therefore, to study the spectral radius of the iteration matrix $M^{-1}N$, where

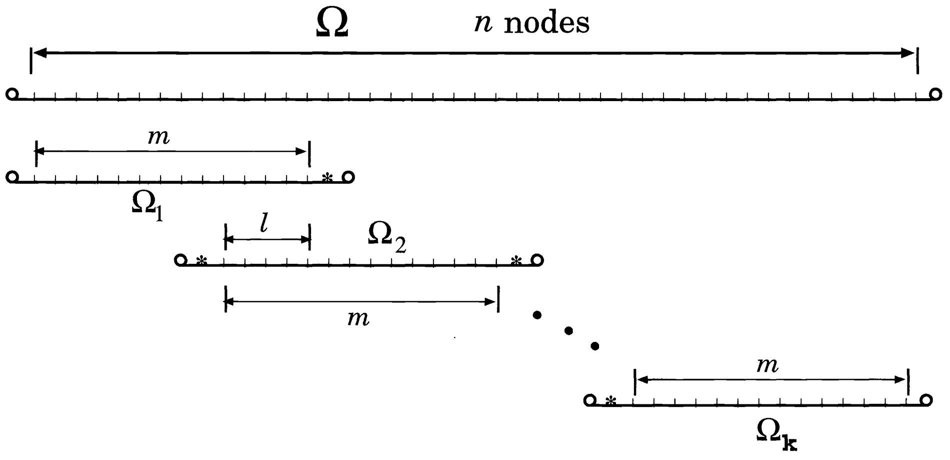


FIG. 3. *1-D overlapping grid.*

$$M = \begin{pmatrix} A_{11} & A_{12} & 0 & 0 & 0 & 0 & 0 \\ A_{21} & B_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & B_2' & A_{23} & 0 & 0 & 0 \\ 0 & 0 & A_{32} & A_{33} & A_{34} & 0 & 0 \\ 0 & 0 & 0 & A_{43} & B_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & B_4' & A_{45} \\ 0 & 0 & 0 & 0 & 0 & A_{54} & A_{55} \end{pmatrix}, \tag{18}$$

$$N = -\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & C_2 & A_{23} & 0 & 0 & 0 \\ A_{21} & C_2' & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & C_4 & A_{45} \\ 0 & 0 & 0 & A_{43} & C_4' & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{19}$$

The OSAM procedure for (2) can also be expressed in the Schwarz splitting form. First, the derivation for a one-dimensional (1-D) model problem is presented. Although the one-dimensional case has no practical importance, it will, however, be used for a higher-dimensional case. Consider

$$-y'' + qy = f, \qquad x \in (0,1), \tag{20}$$

where $q(x) \geq 0$. Denote the number of total nodes as n and the number of overlapping subdomains as k. For convenience, we assume the overlap pattern is uniform. Each subdomain and overlapping part contains m and l nodes, respectively, excluding the boundary layer, which is one grid in width. We also assume that the artificial boundary for each subdomain corresponds to one grid and that no three subdomains have a common overlap (see Figure 3). The open circles represent the artificial boundaries, and the asterisks represent the other grids of the boundary layer for each subdomain.

After discretization using the central finite difference scheme, the resulting linear algebraic equation from (20) is

$$Ax = b, \tag{21}$$

where A is a tridiagonal matrix with p $(p \geq 2)$ on the diagonal and -1 on the off-diagonals. For readability, only the three-subdomain case is displayed.

The partitioned form of A is, then,

$$(22)\quad A = \left(\begin{array}{ccccccccccccccc}
p & -1 & & & & & & & & & & & & & \\
 & \ddots & & & & & & & & & & & & & \\
 & -1 & p & -1 & & & \leftarrow & & & & & & & & \\
 & & -1 & p & -1 & & & & & & & & & & \\
 & & & & \ddots & & & & & & & & & & \\
 & & & & -1 & p & -1 & & & & & & & & \\
 & & & \rightarrow & & -1 & p & -1 & & & & & & & \\
 & & & & & & & \ddots & & & & & & & \\
 & & & & & & & -1 & p & -1 & & \leftarrow & & & \\
 & & & & & & & & -1 & p & -1 & & & & \\
 & & & & & & & & & & \ddots & & & & \\
 & & & & & & & & & & -1 & p & -1 & & \\
 & & & & & & & & & \rightarrow & & -1 & p & -1 & \\
 & & & & & & & & & & & & & \ddots & \\
 & & & & & & & & & & & & & -1 & p
\end{array}\right).$$

The rows which correspond to the inner boundary layer nodes are indicated by "←" or "→". The reconstruction of the boundary layer procedure can be described as follows. First, we multiply the rows $(m-l)$, $(m+1)$, $2(m-l)$, and $(2m-l+1)$ (i.e., those rows corresponding to the inner boundary layer nodes) by $\eta = \theta/p$ and add to rows $m-l+1$, m, $2(m-l)+1$, and $2m-l$, respectively. If the same operations are carried out for the right-hand side, the solution of the new matrix equation $A'x' = b'$ will be the same as that of $Ax = b$, since A' is row equivalent to A. A' is as follows:

$$A' = \left(\begin{array}{ccccccccccccccc}
p & -1 & & & & & & & & & & & & & \\
 & \ddots & & & & & & & & & & & & & \\
 & -1 & p & -1 & & & & & & & & & & & \\
 & -\eta & \theta\text{-}1 & p' & -1 & & & & & & & & & & \\
 & & & & \ddots & & & & & & & & & & \\
 & & & & -1 & p' & \theta\text{-}1 & -\eta & & & & & & & \\
 & & & & & -1 & p & -1 & & & & & & & \\
 & & & & & & & \ddots & & & & & & & \\
 & & & & & & & -1 & p & -1 & & & & & \\
 & & & & & & & -\eta & \theta\text{-}1 & p' & -1 & & & & \\
 & & & & & & & & & & \ddots & & & & \\
 & & & & & & & & & & -1 & p' & \theta\text{-}1 & -\eta & \\
 & & & & & & & & & & & -1 & p & -1 & \\
 & & & & & & & & & & & & & \ddots & \\
 & & & & & & & & & & & & & -1 & p
\end{array}\right),$$

(23)

where $p' = p - \eta$. This step is equivalent to a half-step of the elimination of the boundary layer, namely the weighted combination for the boundary layer nodes. The overlapping block matrices A_{22} and A_{44} in (22) are now modified to

$$A_{22} = A_{44} = \begin{pmatrix} \mathrm{p} & \text{-1} & 0 & \cdots & 0 \\ \text{-1} & \mathrm{p} & \text{-1} & \cdots & 0 \\ & & \ddots & & \\ 0 & \cdots & \text{-1} & \mathrm{p} & \text{-1} \\ 0 & \cdots & 0 & \text{-1} & \mathrm{p} \end{pmatrix} \longrightarrow \begin{pmatrix} p' & \text{-1} & 0 & \cdots & 0 \\ \text{-1} & \mathrm{p} & \text{-1} & \cdots & 0 \\ & & \ddots & & \\ 0 & \cdots & \text{-1} & \mathrm{p} & \text{-1} \\ 0 & \cdots & 0 & \text{-1} & p' \end{pmatrix} = A'_{22} = A'_{44}.$$

To implement the Neumann boundary condition on the inner boundary layers, the GSS is applied. The above overlapping block matrices are split into

$$(24) \qquad \begin{cases} A'_{22} = B_2 + C_2 = C'_2 + B'_2, \\ A'_{44} = B_4 + C_4 = C'_4 + B'_4, \end{cases}$$

where

$$B_2 = B_4 = \begin{pmatrix} p' & \text{-1} & 0 & \cdots & 0 \\ & & \ddots & & \\ 0 & \cdots & \text{-1} & \mathrm{p} & \text{-1} \\ 0 & \cdots & 0 & \text{-1} & p'\text{-1-}\theta \end{pmatrix}, \quad C_2 = C_4 = \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 \\ & & \ddots & & \\ 0 & \cdots & 0 & 0 & 0 \\ 0 & \cdots & 0 & 0 & 1\text{+}\theta \end{pmatrix},$$

$$B'_2 = B'_4 = \begin{pmatrix} p'\text{-1-}\theta & \text{-1} & 0 & \cdots & 0 \\ \text{-1} & \mathrm{p} & \text{-1} & \cdots & 0 \\ & & \ddots & & \\ 0 & \cdots & 0 & \text{-1} & p' \end{pmatrix}, \quad C'_2 = C'_4 = \begin{pmatrix} 1\text{+}\theta & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ & & \ddots & & \\ 0 & \cdots & 0 & 0 & 0 \end{pmatrix}.$$

The diagonal entry p' of the overlapping block matrices A'_{22} and A'_{44} is split into

$$p' = (p' - 1 + \theta) + (1 - \theta) = \tilde{p} + (1 - \theta),$$

where

$$(25) \qquad \tilde{p} = p' - 1 + \theta = p - (\theta/p + (1 - \theta)).$$

Note the right-most side in (25). The two terms, θ/p and $(1 - \theta)$, have been subtracted from the diagonal entry p of the original matrix. These two terms reflect the contributions of two conditions on the inner boundary layer, namely the difference operator on the inner boundary layer and the Neumann boundary condition. This is different from the GSAM, where the splitting only contains the Robin condition.

Then the enhanced form of A', denoted by $\tilde{A}$, is

$$\tilde{A} = \left(\begin{array}{ccccccccccccccccc}
\ddots &&&&&&&&&&&&&&&& \\
-1 & \mathrm{p} & -1 &&&&&&&&&&&&&& \\
-\eta & \theta\text{-}1 & p' & -1 &&&&&&&&&&&&& \\
&&& \ddots &&&&&&&&&&&&& \\
&&& -1 & \tilde{p} &&& 1\text{-}\theta & \theta\text{-}1 & -\eta &&&&&&& \\
-\eta & \theta\text{-}1 & 1\text{-}\theta &&& \tilde{p} & -1 &&&&&&&&&& \\
&&&&&& \ddots &&&&&&&&&& \\
&&&&&& -1 & p' & \theta\text{-}1 & -\eta &&&&&&& \\
&&&&&&&& \ddots &&&&&&&& \\
&&&&&&& -\eta & \theta\text{-}1 & p' & -1 &&&&&& \\
&&&&&&&&&& \ddots &&&&&& \\
&&&&&&&&&& -1 & \tilde{p} &&& 1\text{-}\theta & \theta\text{-}1 & -\eta \\
&&&&&&& -\eta & \theta\text{-}1 & 1\text{-}\theta &&& \tilde{p} & -1 &&& \\
&&&&&&&&&&&&& \ddots &&& \\
&&&&&&&&&&&&& -1 & p' & \theta\text{-}1 & -\eta \\
&&&&&&&&&&&&&& -1 & \mathrm{p} & -1 \\
&&&&&&&&&&&&&&&& \ddots
\end{array}\right).$$

Using the same convention as in [18], matrix A' is equivalent to its enhanced form $\tilde{A}$ under splitting (24) if and only if matrices $B_2 - C'_2$ and $B_4 - C'_4$ are nonsingular according to Theorem 3 in [18]. It will be shown later that this is true if $0 \leq \theta \leq 1$. The diagonal blocks in the above matrix $\tilde{A}$ are not the regular tridiagonal form after the splitting, and are not convenient for the convergence analysis. However, the regular tridiagonal form on the diagonal can be easily recovered. We multiply rows $m - l$, $m + l + 1$, $2m - l$, and $2m + l + 1$ by $(-\eta)$ and add to rows $m - l + 1$, $m + l$, $2m - l + 1$, and $2m + l$, respectively. This results in a new matrix, denoted by $\tilde{A}'$:

$$\tilde{A}' = \left(\begin{array}{ccccccccccccccccc}
\ddots &&&&&&&&&&&&&&&& \\
-1 & \mathrm{p} & -1 &&&&&&&&&&&&&& \\
& -1 & \mathrm{p} & -1 &&&&&&&&&&&&& \\
&&& \ddots &&&&&&&&&&&&& \\
&&& -1 & \tilde{p} &&& 1\text{-}\theta & \theta\text{-}1 & -\eta &&&&&&& \\
-\eta & \theta\text{-}1 & 1\text{-}\theta &&& \tilde{p} & -1 &&&&&&&&&& \\
&&&&&& \ddots &&&&&&&&&& \\
&&&&&& -1 & \mathrm{p} & -1 &&&&&&&& \\
&&&&&&&& \ddots &&&&&&&& \\
&&&&&&&& -1 & \mathrm{p} & -1 &&&&&& \\
&&&&&&&&&& \ddots &&&&&& \\
&&&&&&&&&& -1 & \tilde{p} &&& 1\text{-}\theta & \theta\text{-}1 & -\eta \\
&&&&&&& -\eta & \theta\text{-}1 & 1\text{-}\theta &&& \tilde{p} & -1 &&& \\
&&&&&&&&&&&&& \ddots &&& \\
&&&&&&&&&&&&& -1 & \mathrm{p} & -1 & \\
&&&&&&&&&&&&&& -1 & \mathrm{p} & -1 \\
&&&&&&&&&&&&&&&& \ddots
\end{array}\right),$$

which is now equivalent to matrix $\tilde{A}$. We call $\tilde{A}'$ an *enhanced* OSAM matrix of A.

Let $T_m(p)$ be the m-dimensional tridiagonal matrix with p on the diagonal and -1 on the off-diagonals. Denote by $T_m^l(p, \beta)$ and $T_m^f(p, \beta)$ the two matrices which are identical to $T_m(p)$, except for the last diagonal entry of $T_m^l(p, \beta)$ and the first diagonal entry of $T_m^f(p, \beta)$, which are $p - \beta$. Similarly, $T_m^b(p, \beta)$ denotes a matrix which is identical to $T_m(p)$, except that both the first and last diagonal entries are $p - \beta$. Then the OSAM enhanced matrix $\tilde{A}'$ can be represented as

$$\tilde{A}' = \begin{pmatrix} T_m^l(p, \beta) & -U & 0 \\ -L & T_m^b(p, \beta) & -U \\ 0 & -L & T_m^f(p, \beta) \end{pmatrix},$$

where

$$U = \begin{pmatrix} 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & \cdots & \theta\text{-}1 & 1\text{-}\theta & \eta & \cdots & 0 \end{pmatrix}$$

is an m by m matrix with only three possible nonzero entries located on the lth, $(l+1)$th, and $(l+2)$th columns of the last row. Matrix L can be obtained by rotating U 180°. The convergence behavior of the OSAM depends on the spectral radius of the Jacobi matrix $J = M^{-1}N$, where

$$M = \begin{pmatrix} T_m^l(p, \beta) & 0 & 0 \\ 0 & T_m^b(p, \beta) & 0 \\ 0 & 0 & T_m^f(p, \beta) \end{pmatrix}, \qquad N = \begin{pmatrix} 0 & U & 0 \\ L & 0 & U \\ 0 & L & 0 \end{pmatrix}.$$

For $T_n(p)$, we have [8]

$$\det(T_n(p)) = \begin{cases} \sinh(n+1)\omega/\sinh\omega, & p > 2, \quad 2\cosh\omega = p, \\ n+1, & p = 2, \\ \sin(n+1)\omega/\sin\omega, & p < 2, \quad 2\cos\omega = p. \end{cases}$$

Thus, the following result can be obtained by using the expansion of the determinant of the last (or first) row.

LEMMA 4.1. *Let $T_m^l(p, \beta)$, $T_m^b(p, \beta)$, and $T_m^f(p, \beta)$ be defined as above; then*

$$\det(T_m^l(p, \beta)) = \det(T_m^f(p, \beta)) = \begin{cases} (m+1) - \beta m, & p = 2, \\ (\sinh(m+1)\omega - \beta\sinh m\omega)/\sinh\omega, & p > 2, \end{cases}$$

$$\det(T_m^b(p, \beta)) = \begin{cases} (m+1) - 2\beta m + \beta^2(m-1), & p = 2, \\ (\sinh(m+1)\omega - 2\beta\sinh m\omega + \beta^2\sinh(m-1)\omega)/\sinh\omega, & p > 2. \end{cases}$$

Let

$$t^{(l)} = (t_1^{(l)}, t_2^{(l)}, \cdots, t_m^{(l)})^T, \qquad t^{(b)} = (t_1^{(b)}, t_2^{(b)}, \cdots, t_m^{(b)})^T$$

be the *last* columns of the matrices $(T_m^l(p,\beta))^{-1}$, $(T_m^b(p,\beta))^{-1}$, respectively, and

$$\tilde{t}^{(b)} = (\tilde{t}_1^{(b)}, \tilde{t}_2^{(b)}, \cdots, \tilde{t}_m^{(b)})^T, \qquad \tilde{t}^{(f)} = (\tilde{t}_1^{(f)}, \tilde{t}_2^{(f)}, \cdots, \tilde{t}_m^{(f)})^T$$

be the *first* columns of the matrices $(T_m^b(p,\beta))^{-1}$, $(T_m^f(p,\beta))^{-1}$, respectively. It is not difficult to see that

$$t_j^{(b)} = \tilde{t}_{m-j+1}^{(b)}, \qquad t_j^{(l)} = \tilde{t}_{m-j+1}^{(f)},$$

and

$$(26) \quad t_i^{(l)} = \det(T_{i-1}(p))/\det(T_m^l(p,\beta)) = \begin{cases} i/\Delta_m^l(2,\beta), & p = 2, \\ \sinh i\omega/\Delta_m^l(p,\beta), & p > 2, \end{cases}$$

$$(27) \quad t_i^{(b)} = \det(T_{i-1}^l(p,\beta))/\det(T_m^b(p,\beta)) = \begin{cases} \Delta_{i-1}^l(2,\beta)/\Delta_m^b(2,\beta), & p = 2, \\ \Delta_{i-1}^l(p,\beta)/\Delta_m^b(p,\beta), & p > 2, \end{cases}$$

where

$$\Delta_m^l(p,\beta) = \begin{cases} \det(T_m^l(2,\beta)), & p = 2, \\ \det(T_m^l(p,\beta))\sinh\omega, & p > 2, \end{cases}$$

$$\Delta_m^b(p,\beta) = \begin{cases} \det(T_m^b(2,\beta)), & p = 2, \\ \det(T_m^b(p,\beta))\sinh\omega, & p > 2, \end{cases}$$

and $\det(T_0(p)) = \det(T_0^l(p,\beta)) = 1$.

Applying a transformation similar to that used in [18], the Jacobi matrix J has the same nonzero eigenvalues as the following matrix G:

$$G = \begin{pmatrix} 0 & g_1 & 0 & \\ g_2 & 0 & 0 & g_3 \\ g_3 & 0 & 0 & g_2 \\ 0 & 0 & g_1 & 0 \end{pmatrix},$$

where

$$(28) \quad \begin{aligned} g_1 &= \theta/pt_{m-l-1}^{(l)} + (1-\theta)t_{m-l}^{(l)} - (1-\theta)t_{m-l+1}^{(l)}, \\ g_2 &= \theta/pt_{m-l-1}^{(b)} + (1-\theta)t_{m-l}^{(b)} - (1-\theta)t_{m-l+1}^{(b)}, \\ g_3 &= \theta/pt_{l+2}^{(b)} + (1-\theta)t_{l+l}^{(b)} - -\theta)t_l^{(b)}. \end{aligned}$$

Matrix G has four eigenvalue values, which are:

$$\lambda_{1,2} = \pm\sqrt{g_1(g_2 - g_3)}, \qquad \lambda_{3,4} = \pm\sqrt{g_1(g_2 + g_3)}.$$

Substituting (26) into (28), we have

$$g_1 = \begin{cases} (m-l-\beta(m-l+1))/\Delta^l_m(2,\beta), & p=2, \\ (\sinh(m-l)\omega - \beta\sinh(m-l+1)\omega)/\Delta^l_m(p,\beta), & p>2, \end{cases}$$

$$g_2 = \begin{cases} (1-\beta)^2(m-l)/\Delta^b_m(2,\beta), & p=2, \\ (1-p\beta+\beta^2)\sinh(m-l)\omega/\Delta^b_m(p,\beta), & p>2, \end{cases}$$

$$g_3 = \begin{cases} ((l+1)-2\beta l+\beta^2(l-1))/\Delta^b_m(2,\beta), & p=2, \\ (\sinh(l+1)\omega - 2\beta\sinh l\omega + \beta^2\sinh(l-1)\omega)/\Delta^b_m(p,\beta), & p>2. \end{cases}$$

In particular, when

$$\beta = \begin{cases} (m-l)/(m-l+1), & p=2, \\ \sinh(m-l)\omega/\sinh(m-l+1)\omega, & p>2, \end{cases}$$

we have $g_1 = 0$. Only one iteration is needed for the three-subdomain case. Generally, for any k subdomains, the Jacobi matrix J has the same nonzero eigenvalues as the following $2(k-1)$ by $2(k-1)$ matrix G_k,

$$G_k = \left(\begin{array}{ccccccccccc} 0 & g_1 & 0 & & & & & & & & \\ g_2 & 0 & 0 & g_3 & & & & & & & \\ g_3 & 0 & 0 & g_2 & & & & & & & \\ & & g_2 & 0 & 0 & g_3 & & & & & \\ & & g_3 & 0 & 0 & g_2 & & & & & \\ & & & & \ddots & 0 & 0 & \ddots & & & \\ & & & & & & & g_2 & 0 & 0 & g_3 \\ & & & & & & & g_3 & 0 & 0 & g_2 \\ & & & & & & & & 0 & g_1 & 0 \end{array} \right).$$

In order to obtain a consistent solution from the OSAM with the original problem (22), it is necessary to show that the *enhanced* OSAM matrix $\tilde{A}'$ is equivalent to matrix A.

THEOREM 4.2. *For the model problem, the matrix A is equivalent to its enhanced* OSAM *matrix $\tilde{A}'$, for* $0 \le \theta \le 1$.

Proof. Since A and A', $\tilde{A}$ and $\tilde{A}'$ are row equivalent, we only need to prove A' is equivalent to $\tilde{A}$. From the GSS corresponding to (24), we have

$$B_2 - C_2' = B_4 - C_4' = T^b_l(p,\beta),$$

where $0 < \beta = \theta/p + 1 - \theta \le 1$ for $0 \le \theta \le 1$. From Lemma 4.1, $\det(B_2 - C_2') \ne 0$. So $\tilde{A}$ is equivalent to A' by Theorem 3 in [18]. This completes the proof. □

In a two-dimensional case, the resulting linear algebraic equation is a block version of the corresponding one-dimensional case. Suppose the solution domain has an n^2 grid and is decomposed in the y-direction into three strips. Each subdomain has n by m grids, excluding the artificial boundaries and boundary layers, and all the other assumptions are the same as in the one-dimensional case.

The partitioned coefficient matrix is

$$A=\left(\begin{array}{ccccccccccccccc}
P & -I & & & & & & & & & & & & & \\
 & \ddots & & & & & & & & & & & & & \\
 & -I & P & -I & & & \leftarrow & & & & & & & & \\
 & & -I & P & -I & & & & & & & & & & \\
 & & & & \ddots & & & & & & & & & & \\
 & & & & -I & P & -I & & & & & & & & \\
 & & \rightarrow & & & -I & P & -I & & & & & & & \\
 & & & & & & & \ddots & & & & & & & \\
 & & & & & & & -I & P & -I & & & \leftarrow & & \\
 & & & & & & & & -I & P & -I & & & & \\
 & & & & & & & & & & \ddots & & & & \\
 & & & & & & & & & & -I & P & -I & & \\
 & & & & & & & & \rightarrow & & & -I & P & -I & \\
 & & & & & & & & & & & & & \ddots & \\
 & & & & & & & & & & & & & -I & P
\end{array}\right)$$

(29)

where I is an n by n identity matrix and $P = T_n(p)$ with $p = 4$ for the model problem. Similarly, the rows which correspond to the inner boundary layer nodes are indicated by " $\leftarrow$ " or "$\rightarrow$". After the splitting of the overlapping blocks, the final *enhanced* OSAM matrix is

$$\tilde{A}'=\left(\begin{array}{ccccccccccccccccccc}
P & -I & & & & & & & & & & & & & & & & & \\
 & \ddots & & & & & & & & & & & & & & & & & \\
 & -I & P & -I & & & & & & & & & & & & & & & \\
 & & -I & p & -I & & & & & & & & & & & & & & \\
 & & & & \ddots & & & & & & & & & & & & & & \\
 & & & & -I & \tilde{P} & & & \Gamma & -\Gamma & -\Theta & & & & & & & & \\
 & -\Theta & -\Gamma & \Gamma & & & \tilde{P} & -1 & & & & & & & & & & & \\
 & & & & & & & \ddots & & & & & & & & & & & \\
 & & & & & & & -I & P & -I & & & & & & & & & \\
 & & & & & & & & & \ddots & & & & & & & & & \\
 & & & & & & & & & -I & P & -I & & & & & & & \\
 & & & & & & & & & & & \ddots & & & & & & & \\
 & & & & & & & & & & & -I & \tilde{P} & & & \Gamma & -\Gamma & -\Theta & \\
 & & & & & & & & -\Theta & -\Gamma & \Gamma & & & \tilde{P} & -I & & & & \\
 & & & & & & & & & & & & & & \ddots & & & & \\
 & & & & & & & & & & & & & & -I & P & -I & & \\
 & & & & & & & & & & & & & & & -I & P & -I & \\
 & & & & & & & & & & & & & & & & & \ddots & \\
 & & & & & & & & & & & & & & & & & -I & P
\end{array}\right),$$

where $\Theta = \eta I$ with $\eta = \theta/p$,

$$(30) \qquad \tilde{P} = P - (\Theta + \Gamma), \qquad \Gamma = \begin{pmatrix} 1-\theta & \eta & & & \\ \eta & 1-\theta & \eta & & \\ & & \ddots & & \\ & & \eta & 1-\theta & \eta \\ & & & \eta & 1-\theta \end{pmatrix}.$$

As in the one-dimensional case, Θ and Γ are subtracted from P to reflect the contributions of both the differential operator and the Neumann boundary condition on the boundary layer. The difference between the OSAM and the GSAM is now clear. Notice that the matrix Γ in (30) is a tridiagonal matrix. For GSAM the corresponding splitting is $P = (P - \Gamma') + \Gamma'$, and Γ' is a diagonal matrix. $\Gamma' = \text{diag}(\alpha)$, where $\alpha = (1-\omega)/(1-\omega+\omega h)$, and h represents the mesh size. The difference in a three-dimensional case is also not difficult to see.

The splitting matrices M and N of $\tilde{A}'$ have the same block sparsity pattern as in a one-dimensional case. They are the block version of the corresponding matrices in the one-dimensional case. Each block is either an identity or a tridiagonal matrix. The Jacobi iterative matrix of the OSAM $J = M^{-1}N$ is also similar. Let E_n be a matrix in which each column corresponds to a normalized eigenvector of matrix $T_n(p)$. Then E_n is an orthogonal matrix such that $E_n^T\, T_n(p)\, E_n = \text{diag}\{p_i\},\ p_i = p - 2\cos(i\pi/(n+1)),\ i = 1, 2, \ldots, n$.

Let

$$U = \begin{pmatrix} I_m \otimes E_n & 0 & 0 \\ 0 & I_m \otimes E_n & 0 \\ 0 & 0 & I_m \otimes E_n \end{pmatrix};$$

then J and $J' = U^T J U$ are similar:

$$\begin{aligned} J' &= U^T J U = U^T(M^{-1}N)U = (U^T M^{-1} U)(U^T N U) \\ &= (U^T M U)^{-1}(U^T N U) = \widetilde{M}^{-1}\,\widetilde{N}. \end{aligned}$$

Matrices $\widetilde{M}$ and $\widetilde{N}$ are exactly the same as M and N, except that those tridiagonal matrices are changed to diagonal matrices as follows:

$$\begin{aligned} &\Gamma \to \text{diag}\{\gamma_i\}, \quad \gamma_i = (1-\theta) + 2\eta\cos\omega_i, \\ &P \to \text{diag}\{p_i\}, \quad p_i = p - 2\cos\omega_i, \\ &\widetilde{P} \to \text{diag}\{\tilde{p_i}\}, \quad \tilde{p_i} = (p - \eta - 1 + \theta) - 2(1+\eta)\cos\omega_i, \end{aligned}$$

where $\omega_i = i\pi/(n+1)$.

The above matrix J' can be spectrally decomposed into n one-dimensional Jacobi iterative matrices. Let Q be the permutation matrix which permutes row $(k-1)n + i$ to $(i-1)3m + k$, $k = 1, \ldots, 3m$, $i = 1, \ldots, n$. Then we have

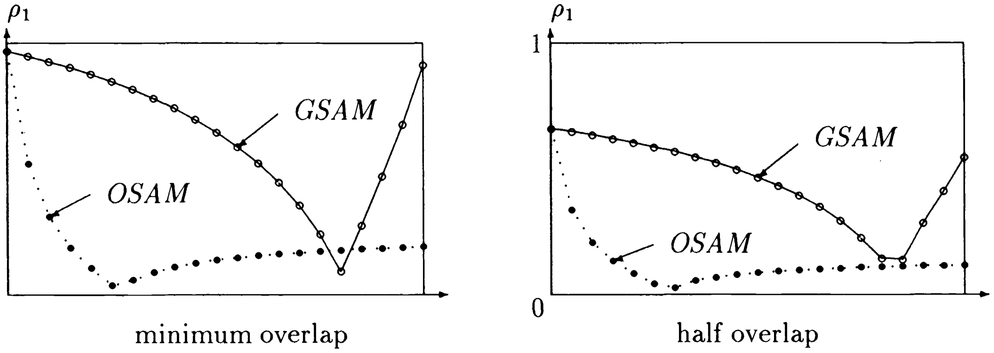


FIG. 4. *2-D three-subdomain case.*

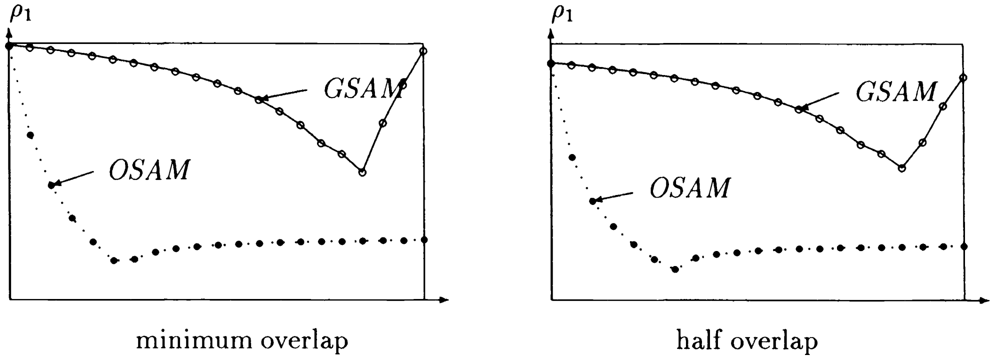


FIG. 5. *2-D seven-subdomain case.*

$$Q^T J' Q = Q^T(\widetilde{M}^{-1}\widetilde{N})Q = (Q^T\widetilde{M}Q)^{-1}(Q^T\widetilde{N}Q)$$

$$= \begin{pmatrix} M_1^{-1} & & & \\ & M_2^{-1} & & \\ & & \ddots & \\ & & & M_n^{-1} \end{pmatrix} \begin{pmatrix} N_1 & & & \\ & N_2 & & \\ & & \ddots & \\ & & & N_n \end{pmatrix}.$$

Each diagonal block $M_i^{-1}N_i$ is a $3m$ by $3m$ matrix, which is a Jacobi iterative matrix for the corresponding one-dimensional case. As in the analysis in GSAM [18], the largest eigenvalue of J' is decided by the maximum eigenvalue of $M_1^{-1}N_1$.

For the model problem in the two-dimensional (2-D) case, the maximum eigenvalues ρ_1 versus the parameter θ for the OSAM are calculated and shown in Figures 4 and 5. For comparison, the maximum eigenvalues, still denoted by ρ_1, versus the parameter ω for the GSAM are also shown there. Both parameters range from 0 to 1. The maximum eigenvalues are between 0 and 1. For a minimum overlapping case, the total number of grids n in each direction is 58 for 3 subdomains or 136 for 7 subdomains. For the corresponding half-overlapping case, the total number of grids varies slightly to fit the decomposition.

It can be seen from Figures 4 and 5 that the OSAM gives better convergence behavior. For a wide range of the parameter θ, the OSAM is better than the GSAM and the SAM, which is the special case of the GSAM with $\omega = 1$. Sensitivity of convergence of the OSAM to its parameter is almost insignificant. Moreover, the OSAM with a minimum overlap is still much better than the SAM with a half-overlap. As the number of subdomains increases, the improvement is even more significant.

5. Numerical tests. Results for several testing problems in a 2-D case are presented in this section. All the tests are carried out on a Sun 4/670 server, which is normally rated at four MFLOPS. The arithmetic is performed in Fortran 77 double precision. The differential equations are discretized by the standard central difference scheme. The upwind scheme has also been tested. The convergence characteristics are similar to those of the central scheme. To save space, these results will not be listed. Except for the first problem, all the rest are nonself-adjoint asymmetric problems. There is one discontinuous coefficient problem and one indefinite problem.

For each test problem, the solution domain is decomposed into a different number of subdomains, such as 4 by 4, 10 by 10, etc. In all the cases, each subdomain contains a 20 by 20 grid of unknowns and the minimum overlap is considered. Thus, the total number of unknowns will be increased as the number of subdomains increases. We are mainly concerned with the improvement of the OSAM over the SAM and the reduction of the sensitivity of the convergence behavior to the parameter.

The domain decomposition method is used as a preconditioner, and Bi-CGSTAB is employed for the acceleration scheme. The convergence behavior for other acceleration schemes is similar. The iteration is stopped if the l_2 norm of the residual is less than 10^{-5} times the initial residual norm.

There are several notations in the tables of results. The SAM represents the result for using the traditional Schwarz alternating method as the preconditioner. The "$*$" and "$**$" mean that the OSAM does not converge within 100 and 200 iterations, respectively, and the iteration is stopped. "Iter" and "SubD" represent the number of linear iterations needed to reach the precision and the total number of subdomains, respectively. We also define the improvement factor to be $\tau = \frac{\text{SAM}-\text{OSAM}}{\text{SAM}}$. The notation τ_b represents the best improvement factor of the results.

5.1. Helmholz equation. In this test, the following problem is considered:

$$\begin{cases} -\Delta u + u &= f, \quad x \in \Omega, \\ u &= g, \quad x \in \partial\Omega, \end{cases}$$

with $\Omega = (0,1) \times (0,1)$ and true solution $u = e^{x+y} \sin(2x) \cos y$.

To verify our convergence analysis, the problem is solved using pure domain decomposition first. The solution domain is decomposed into three strips. The number of iterations versus parameter is shown in Figure 6. The two curves indicated by "osam.comp" and "gsam.comp" represent the number of iterations (from numerical testing) versus the parameter ω for the GSAM, and θ for the OSAM, respectively. The other two curves, "osam.anal" and "gsam.anal," are the number of iterations needed, which are calculated analytically from our spectral results. The numerical tests match our analysis very closely.

For the rest of the tests, the domain decomposition technique is applied as a preconditioner. The number of iterations is much decreased. Furthermore, the sensitivity to the parameter is also reduced. The corresponding results with the same decomposition pattern as for the above test are shown in Figure 7. The advantages of using the domain decomposition method as a preconditioner are obvious.

The more general case is that in which the solution domain is decomposed in both directions. Table 1(a) gives the result of the Helmholz equation in this case. From Table 1(a), it can be seen that a small parameter value θ is not good for the OSAM. In this case, the Neumann

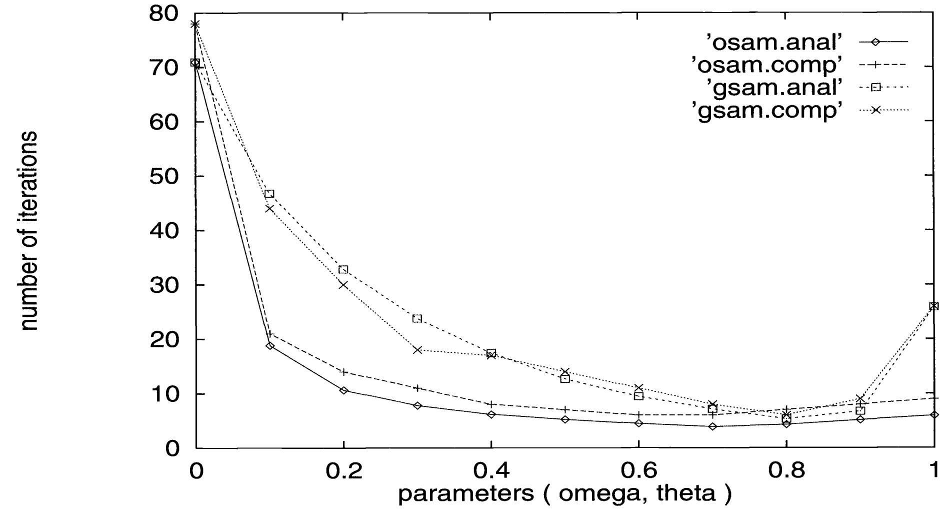


FIG. 6. *Number of iterations (analytical and computational).*

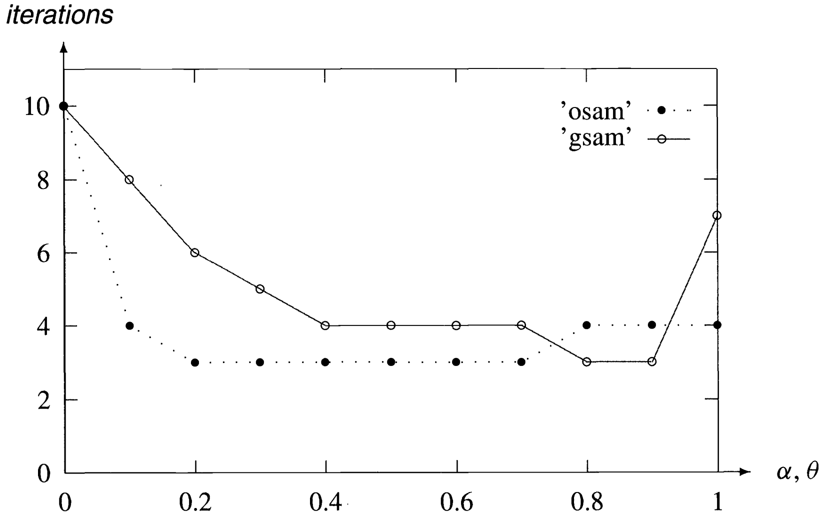


FIG. 7. *Number of iterations: three strips with domain decomposition as preconditioner.*

condition is more dominant than the differential operator on the boundary layer, which means the Neumann condition shares more "weight" in the weighted combination. The Neumann condition is bad, as we have already shown in the previous analysis. But for $0.2 < \theta \leq 1$, the OSAM is an obvious improvement over the SAM. There is little sensitivity shown for $\theta > 0.2$. Note that the improvement in performance and the reduction of the sensitivity to the parameter θ exist not only for the model problem, but also for the rest of our tests. Moreover, the OSAM improves more with difficult problems.

5.2. Stress in helical spring. If a single turn of a helical spring of small angle α and radius R is deformed into a plane ring under the influence of an axial load, the stress-function Φ can be shown to satisfy the differential equation [6]

$$\Phi_{xx} + \Phi_{yy} + \frac{3}{R - y}\Phi_x - 2G\lambda = 0,$$

TABLE 1

Helmholz equations

SubD	10×10	8×8	4×4
θ	Iter	Iter	Iter
0.1	*	79	37
0.2	54	37	15
0.3	13	12	7
0.4	12	10	6
0.5	12	10	6
0.6	12	10	6
0.7	12	10	6
0.8	13	10	6
0.9	13	11	6
1.0	13	11	7
SAM	22	18	10
τ_b	0.455	0.444	0.400

(a)

Stress in helical spring

SubD	8×12	6×8	3×4
θ	Iter	Iter	Iter
0.1	68	42	18
0.2	17	12	7
0.3	14	11	6
0.4	16	13	6
0.5	17	14	6
0.6	17	13	6
0.7	18	14	7
0.8	20	15	7
0.9	20	15	7
1.0	19	14	9
SAM	29	20	10
τ_b	0.517	0.450	0.400

(b)

and it vanishes on the boundary Γ, where Γ is the boundary of the cross-section in the (x, y)-plane which contains the axis of the spring. G is the modulus of rigidity and $\lambda = \sin\alpha\cos\alpha/R$.

The special case we considered is that in which the problem has rectangular cross-section $\Omega = (-.5, .5) \times (-1, 1)$ and $R = 5$. The problem has an exact solution [14],

$$\Phi = (1 - y^2)(1 - 4x^2)(5 - y^3)(0.0004838y + 0.0010185).$$

The angle α is specified for a fixed modulus of rigidity G. In this case, the solution domain is no longer a square. We test the OSAM for the case of different mesh lengths in the two directions. The solution domain is decomposed into 3 by 4, 6 by 8, and 8 by 12 subdomains, respectively.

Unlike the previous one, the results of this problem show that the OSAM performs better than the SAM with only one exception: $\theta = 0.1$. For the optimal case, $\theta = 0.3$, the OSAM takes only half as many iterations as the SAM. The general convergence behavior of the OSAM with respect to the parameter θ follows the same rule as in the last test. Table 1(b) represents the detailed results of this test.

5.3. Variable coefficients of second order derivative terms. In the previous two cases, the second order operator is the Laplacian. This problem has continuous variable coefficients for second order derivative terms. The unknown is defined on a unit square with homogeneous Dirichlet boundary condition. It satisfies the following equation:

$$((1 + x^2)u_x)_x + u_{yy} + (\tan y)^3 u_y = -100x^2.$$

The numerical results were given in Table 2(a). In general, the improvement factor is still a little more than one-third on average, and the best case of the OSAM saves almost half the work of the SAM.

5.4. Discontinuous coefficient problem. In this test, the following equation was considered:

$$(K_x u_x)_x + (K_y u_y)_y + u_x + u_y = \sin(\pi xy).$$

The unknown vanishes on the boundary of the solution domain $\Omega = (0, 1) \times (0, 1)$, where

$$K_x = K_y = \begin{cases} 1, & [0, .5] \times [0, .5], \\ 10^{+3}, & [.5, 1] \times [0, .5], \\ 10^{-3}, & [0, .5] \times [.5, 1], \\ 1, & [.5, 1] \times [.5, 1]. \end{cases}$$

TABLE 2

Variable coefficients

SubD	10×10	8×8	4×4
θ	Iter	Iter	Iter
0.1	$*$	$*$	26
0.2	$*$	$*$	41
0.3	16	13	8
0.4	15	12	7
0.5	17	13	7
0.6	17	14	7
0.7	18	14	7
0.8	19	14	7
0.9	20	16	7
1.0	19	15	8
SAM	28	23	11
τ_b	0.464	0.478	0.364

(a)

Discontinuous coefficients

SubD	10×10	8×8	4×4
θ	Iter	Iter	Iter
0.1	$*$	$*$	$*$
0.2	87	63	$*$
0.3	16	15	10
0.4	16	12	8
0.5	13	11	7
0.6	16	11	7
0.7	17	12	7
0.8	19	11	9
0.9	16	11	9
1.0	17	12	10
SAM	31	22	13
τ_b	0.548	0.500	0.462

(b)

Generally, discontinuous coefficient problems are relatively difficult to solve. The harmonic scheme [1, 12] is applied to discretizing the second order derivative terms in this test. The numerical linear approaches are the same as for the last several tests, which aim to compare the OSAM and the SAM. The results are shown in Table 2(b). It can be seen that the improvement factor in this test is larger than those in the last several ones. From this test, the OSAM shows great potential for difficult problems. This will be further demonstrated by the following test.

5.5. Variable-coefficient, indefinite problem. The problem tested is

$$\begin{aligned} Lu = {} & -[(1+\tfrac{1}{2}\sin(50\pi x))u_x]_x - [(1+\tfrac{1}{2}\sin(50\pi x)\sin(50\pi y))u_y]_y \\ & + 20\sin(10\pi x)\cos(10\pi y)u_x - 20\cos(10\pi x)\sin(10\pi y)u_y + cu, \end{aligned} \tag{31}$$

where $u = \exp(xy)\sin(\pi x)\sin(\pi y)$ is defined on a unit square. This testing problem is taken from the paper of Cai, Gropp, and Keyes [4]. The sign of the coefficient c in (31) has a keen effect on this problem. The difficulty of the linear system depends on both the mesh widths $\triangle x$, $\triangle y$ and the magnitude of c. For a fixed number of grids, the larger the magnitude of the negative value c, the more difficult the problem will be. In this work, the cases for $c = -20$, -70 were tested. We will see that the second case requires much more work than the first one.

The numerical results are reported in Table 3. From these results, the difficulty of this problem is obvious as compared with the corresponding count of iterations of the previous tests. The difficulty is also reflected in the performance of the OSAM with respect to the parameter θ. As $\theta = 0.3$, the OSAM failed to compete with the SAM in the count of iterations for this problem, but the OSAM was better than the SAM in all the previous tests.

When $c = -20$, the problem is more weakly indefinite than for $c = -70$ with the same number of grids. The results show that the work done for $c = -20$ is only a little more than in corresponding situations in the last several tests. The improvement factor varies from one third to one half. However, when $c = -70$, the OSAM demonstrates its superiority over the SAM. The improvement factors for all three decomposition cases are greater than one half. For the 10 by 10 subdomain case, the SAM took 144 iterations, while the OSAM only needed 43 iterations for the optimal parameter. It is more than three times faster than the SAM. Even in the worst case, $\theta = 1$, which is known to not be good from the analysis and tests, the number

TABLE 3
Results for indefinite problems.

SubD	10×10		8×8		4×4	
θ	Iter		Iter		Iter	
	$c=-20$	$c=-70$	$c=-20$	$c=-70$	$c=-20$	$c=-70$
0.1	*	**	*	**	55	**
0.2	*	**	*	**	29	173
0.3	*	**	*	194	18	76
0.4	21	49	18	38	10	29
0.5	20	58	17	31	9	26
0.6	21	46	18	38	9	27
0.7	23	43	18	37	9	21
0.8	22	49	18	33	9	19
0.9	26	60	19	48	10	19
1.0	25	82	18	41	11	19
SAM	37	144	30	85	14	42
τ_b	0.459	0.713	0.433	0.665	0.357	0.548

of iterations is 82 for the OSAM, and the improvement factor in this case is still larger than one third. For an 8 by 8 subdomain case, the SAM took 85 iterations, but the OSAM requires less than half of that with only one exception, $\theta = 0.9$. When $\theta = 0.5$, the OSAM took only 31 iterations. This test further demonstrates that the OSAM performs better for difficult problems.

6. Conclusion. In this paper, a new extension of the classical Schwarz alternating method, the overdetermined Schwarz alternating method, is proposed. In this new approach, a stronger coupling is imposed on the artificial boundary layers. The superior convergence behavior is demonstrated for a wide range of test problems. In particular, the weighted parameter θ does not cause the sensitivity problem from which the GSAM suffers. So far, our testing has been restricted to the single level approach. A multilevel preconditioner approach is a natural future extension of this work.

REFERENCES

[1] J. B. BELL, C. N. DAWSON, AND G. R. SHUBIN, *An unsplit, higher order Godunov method for scalar conservation laws in multiple dimensions*, J. Comput. Phys., 89 (1988), pp. 1–24.

[2] P. BJORSTAD AND M. SKOGEN, *Domain decomposition algorithms of Schwarz type, designed for massively parallel computers*, in Fifth International Symposium on Domain Decomposition Methods for Partial Differential Equations, T. Chan, D. Keyes, G. Meurant, J. Scroggs, and R. Voigt, eds., SIAM, Philadelphia, PA, 1991.

[3] X.-C. CAI, *An additive Schwarz algorithm for nonselfadjoint elliptic equations*, in Third International Symposium on Domain Decomposition Methods for Partial Differential Equations, T. Chan, R. Glowinski, J. Periaux, and O. Widlund, eds., SIAM, Philadelphia, PA, 1990.

[4] X.-C. CAI, W. GROPP, AND D. KEYES, *A comparison of some domain decomposition algorithms for nonsymmetric elliptic problems*, in Fifth International Symposium on Domain Decomposition Methods for Partial Differential Equations, T. Chan, D. Keyes, G. Meurant, J. Scroggs, and R. Voigt, eds., SIAM, Philadelphia, PA, 1992.

[5] T. CHAN AND D. GOVAERTS, *Schwarz-Schur: Overlapping versus nonoverlapping domain decomposition*, Tech. Report, UCLA, Los Angeles, CA, 1988.

[6] L. COLLATZ, *The Numerical Treatment of Differential Equations*, 3rd ed., Springer-Verlag, New York, 1966.

[7] M. DRYJA AND O. WIDLUND, *An additive variant of Schwarz alternating method for the case of many subregions*, Tech. Report 339, New York University, New York, 1987.

[8] C. FISCHER AND R. USMANI, *Properties of some tridiagonal matrices and their application to boundary value problems*, SIAM J. Numer. Anal., 6 (1969), pp. 127–142.

[9] R. GLOWINSKI, Q. V. DINH, AND J. PERIAUX, *Domain decomposition methods for nonlinear problems in fluid dynamics*, Tech. Report, INRIA, Le Chesnay, France, 1980.

[10] L. KANG, *The Schwarz algorithm*, Tech. Report, Wuhan University, Wuhan, China, 1981.

[11] P. L. LIONS, *On the Schwarz alternating method III: A variant for nonoverlapping subdomains*, in Third International Symposium on Domain Decomposition Methods for Partial Differential Equations, T. Chan, R. Glowinski, J. Periaux, and O. Widlund, eds., SIAM, Philadelphia, PA, 1989, pp. 202–216.

[12] A. MAYO, *The fast solution of Poisson's and biharmonic equations on irregular regions*, SIAM J. Numer. Anal., 21, (1984), pp. 285–299.

[13] K. MILLER, *Numerical analogs to the Schwarz alternating procedure*, Numer. Math. 7 (1965), pp. 91–103.

[14] J. RICE AND R. BOISVERT, *Solving Elliptic Problems Using ELLPACK*, Springer-Verlag, New York, 1985.

[15] G. RODRIGUE, *Inner/outer iterative methods and numerical Schwarz algorithms*, J. Parallel Comput., 2 (1985), pp. 205–218.

[16] D. STOUTEMYER, *Numerical Implementation of the Schwarz Alternating Procedure for Elliptic Partial Differential Equations*, Ph.D. thesis, Stanford University, Stanford, CA, 1972.

[17] W.-P. TANG, *Schwarz Splitting and Template Operators*, Ph.D. thesis, Stanford University, Stanford, CA, 1987.

[18] ———, *Generalized Schwarz Splitting*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 573–595.

[19] J. WANG, *Convergence analysis of the Schwarz algorithm and multilevel decomposition iterative methods II: nonselfadjoint and indefinite elliptic problems*, SIAM J. Numer. Anal., 30 (1993), pp. 953–970.

[20] O. WIDLUND, *Some Schwarz methods for symmetric and nonsymmetric elliptic problems*, in Fifth International Symposium on Domain Decomposition Methods for Partial Differential Equations, T. Chan, D. Keyes, G. Meurant, J. Scroggs, and R. Voigt, eds., SIAM, Philadelphia, PA, 1992.

[21] J. XU, *Iterative methods by space decomposition and subspace correction: A unifying approach*, SIAM Rev., 34 (1992), pp. 581–613.

RANDOM RELAXATION OF FIXED-POINT ITERATION*

MARKKU VERKAMA†

Abstract. This paper considers a stochastic fixed-point iteration where each coordinate is updated with a certain probability and otherwise left unchanged. The iteration is interesting from the viewpoint of parallel distributed computation because the realized sequences belong to the class of asynchronous fixed-point iterations. It is demonstrated with a linear system that the convergence conditions for randomly relaxed iterations are less stringent than their asynchronous counterparts, and that they can illuminate the tightness of the convergence conditions for asynchronous iterations, which are typically worst-case conditions.

Key words. asynchronous algorithm, fixed points, stochastic stability, linear systems

AMS subject classifications. 65Y10, 60G35, 65F10, 93E15

1. Introduction.

In this paper we examine the following stochastic iteration in the n-dimensional Euclidean space $\mathbf{R}^n$. For a given function $f : \mathbf{R}^n \mapsto \mathbf{R}^n$, let $x_i(k+1) = f_i(x(k))$ with probability p and let $x_i(k+1) = x_i(k)$ with probability $1-p$. All randomizations across the coordinates and across the iterates are independent in the process. In other words, the coordinates are updated independently at each iteration step and the iterations are independent as well. An iteration of this type will be called a *randomly relaxed* iteration. When the coordinates of x are updated randomly, a sequence of random vectors in $\mathbf{R}^n$ is realized. We are interested in the convergence of the random vector sequences to a fixed point of f.

Randomly relaxed iterations are closely related to so-called asynchronous fixed-point iterations. In the (zero-delay) *asynchronous* execution of the iteration

$$x_i \leftarrow f_i(x_1, \dots, x_n), \quad i = 1, \dots, n, \tag{1.1}$$

the coordinates of x are updated in arbitrary order, each infinitely many times. In so-called *totally* asynchronous iterations the values of the coordinates used on the right side of (1.1) may also be arbitrarily out-of-date, so that $x(k)$ does not sufficiently describe the state of the system [2], [6]. Such iterations also go under the name chaotic relaxation. It is clear that randomly relaxed iterations belong to the class of asynchronous iterations—an arbitrary set of coordinates is updated at a time. Hence the analysis of randomly relaxed iterations can give new insight into the convergence of asynchronous iterations, especially zero-delay iterations. A difference to iterations with delays is that coordinate values older than $x(k)$ cannot appear in the computation of $x(k+1)$ in randomly relaxed iterations.

Tsitsiklis [11] has shown, in a rather general sense, that zero-delay asynchronous iterations converge if and only if there exists a Lyapunov function that testifies to this. Related results were obtained in [3] and [4] for linear systems in $\mathbf{R}^n$. These studies, like most of the studies on asynchronous iterations, are, however, concerned with the notion of convergence that requires all possible sequences $\{x(k)\}$ to converge to a fixed point of f. It is therefore interesting to note that a randomly relaxed iteration can converge, for example, almost surely (that is, with probability 1) even if the necessary conditions of [11] are violated. This can happen if the probability measure over the events that produce diverging sequences is zero. This kind of analysis can give an idea of the rarity of worst-case events. One can thus think that results of the convergence of randomly relaxed iterations shed light on the tightness of the convergence conditions derived for asynchronous iterations.

*Received by the editors January 12, 1994; accepted for publication (in revised form) March 6, 1995. This research has been supported by the Väinö Tanner Foundation, the Finnish Cultural Foundation, and the Swedish Foundation of Culture in Finland.

†Systems Analysis Laboratory, Helsinki University of Technology, Otakaari 1, FIN-02150 Espoo, Finland (markku.verkama@ntc.nokia.com).

To illustrate this we shall consider iterations with affine functions. For these, both necessary and sufficient conditions for convergence under total asynchronism are known [6]. Necessary and sufficient conditions for the quadratic mean (q.m.) convergence of randomly relaxed iterations will be derived here. These conditions turn out to be less stringent than the conditions in [6]. It is also shown that a randomly relaxed iteration can converge in a stochastic sense for some $p < 1$ even if the Jacobi iteration $x(k+1) = f(x(k))$, which corresponds to $p = 1$, is unstable. It thus appears that a certain type of asynchronicity can induce stability into otherwise unstable iterations.

This study is not the first one that uses stochastic models to analyze asynchronous iterations. The effects of random communication delays and processor failures have been considered in [1] and [10], and a Poisson transition model has been used to analyze the effects of different processing times in [8].

We shall use standard vector notation throughout the paper. Subscripts refer to vector and matrix elements. The transpose is denoted by the prime; the unit matrix is I. $\|\cdot\|$ denotes the Euclidean norm, and $\|\cdot\|_w$ denotes weighted maximum norm; that is, $\|x\| = (x'x)^{1/2}$ and $\|x\|_w = \max_i |x_i|/w_i$ $(w_i > 0)$. $\rho(A)$ denotes the spectral radius of a matrix A. $A \otimes B$ denotes the Kronecker product of matrices A and B and it is given by

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1m}B \\ \vdots & \ddots & \vdots \\ a_{n1}B & \cdots & a_{nm}B \end{bmatrix},$$

where $A = (a_{ij})$. The matrix $|A|$ is obtained from A by taking absolute value of all elements of A; hence, $|A| = (|a_{ij}|)$. δ_{ij} denotes the Kronecker delta. Finally, $E[\cdot]$ denotes expected value.

2. Random relaxation. Next we define random relaxation and a convergence concept suitable for the analysis of randomly relaxed iterations.

DEFINITION 2.1. *Let* $\{\xi_i(k); k = 0, 1, \ldots\}$ $(i = 1, \ldots, n)$ *be random sequences such that* $\Pr[\xi_i(k) = 1] = p$ *and* $\Pr[\xi_i(k) = 0] = 1 - p$, *where* $p \in (0, 1]$ *is a constant. Randomly relaxed fixed-point iteration of a function* $f : \mathbf{R}^n \mapsto \mathbf{R}^n$ *is defined by*

$$(2.1) \qquad x_i(k+1) = x_i(k) + \xi_i(k)(f_i(x(k)) - x_i(k)), \quad i = 1, \ldots, n, \quad k = 0, 1, \ldots.$$

Iteration (2.1) defines a sequence of random vectors for a given initial vector $x(0)$. Note that for any k_0, the probability of the event "$\xi_i(k) = 0$ for all $k \geq k_0$" (i.e., of the event that x_i is not updated after k_0) is zero, and that by Borel's zero-one criterion [9, p. 228] all coordinates are updated infinitely many times with probability 1. It is, thus, intuitive that if the iteration (2.1) converges in some sense, it must converge to a fixed point of f. In particular, if $x^* = f(x^*)$ and if $x(k_0) = x^*$ for some k_0, then $x(k) = x^*$ for all $k \geq k_0$.

There is another way of looking at random relaxation. Define the operators $T_j : \mathbf{R}^n \mapsto \mathbf{R}^n$, $j = 1, \ldots, 2^n$, by taking all the possible coordinate combinations of f and the identity operator; that is, either $T_{j,i}(x) := f_i(x)$ or $T_{j,i}(x) := x_i$. Then at each step of the iteration, one of the operators is applied with a certain probability $p^{n_j}(1-p)^{n-n_j}$, where $n_j \in \{0, \ldots, n\}$. This makes explicit the relationship to the zero-delay asynchronous iterations in [11]. When $p = 1$ only the operator f is applied, and iteration (2.1) becomes the Jacobi-type fixed-point iteration. Otherwise, the Jacobi iteration as well as the Gauss–Seidel-type iterations are possible realizations among others.

When we define the diagonal random matrices $\Xi(k) = (\xi_i(k)\delta_{ij})$, (2.1) can be written equivalently as

$$(2.2) \qquad x(k+1) = x(k) + \Xi(k)(f(x(k)) - x(k)).$$

Since $\xi_i(k)$ are identically distributed across k, we will refer to the generic random variables as ξ_i and Ξ; the same convention will be used with other random variables and matrices as well.

In this paper we consider the following concept of convergence.

DEFINITION 2.2. *Let $x(k)$, $k = 1, 2, \dots$, be generated by the iteration* (2.1) *starting from an initial vector $x(0)$. We say that the iteration* (2.1) *converges in q.m. to a fixed point x^* of f if*

$$\lim_{k \to \infty} E[\|x(k) - x^*\|^2] = 0 \tag{2.3}$$

for any $x(0)$.

This definition is adapted from the standard definitions used in probability theory; see, e.g., [7], [9].

3. Some convergence results. Let us now consider iterations with affine functions of the form

$$f(x) = Ax + b, \tag{3.1}$$

where A is an $n \times n$ matrix and b is an n vector. Assume that $I - A$ is nonsingular. Then f has a unique fixed point $x^* = (I - A)^{-1}b$.

It was shown in [6] that totally asynchronous fixed-point iterations of f converge to x^* if and only if

$$\rho(|A|) < 1. \tag{3.2}$$

This is also a sufficient condition for the convergence of zero-delay asynchronous iterations. It is, however, not generally known if and how (3.2) can be relaxed for these iterations. Clearly, a necessary condition is always $\rho(A) < 1$, because otherwise the Jacobi iteration $x(k + 1) = Ax(k) + b$ is unstable.

Next we derive results concerning randomly relaxed iterations with the same function. Using (2.2) we can write the randomly relaxed iteration as

$$x(k + 1) = x(k) + \Xi(k)(A - I)x(k) + \Xi(k)b. \tag{3.3}$$

Inserting $y = x - x^*$ into (3.3) shows that, without loss of generality, we can always consider the iteration $x(k + 1) = \Psi(k)x(k)$ where $\Psi(k) = I + \Xi(k)(A - I)$.

It will be useful to define $\zeta_i(k) = \xi_i(k) - E[\xi_i(k)] = \xi_i(k) - p$ and denote $Z(k) = (\zeta_i(k)\delta_{ij}) = \Xi(k) - pI$. Hence, $E[\zeta_i(k)\zeta_j(l)] = p(1 - p)\delta_{ij}\delta_{kl}$ and $E[Z(k)] = 0$. One can then write

$$\Psi(k) = I + p(A - I) + Z(k)(A - I). \tag{3.4}$$

This shows that the randomly relaxed iteration contains a deterministic part and a stochastic part with a zero expected value (even though zero is not a possible realization). The deterministic part corresponds precisely to the underrelaxed Jacobi iteration $x(k+1) = x(k) + p(f(x(k)) - x(k))$.

A necessary and sufficient condition for the convergence of (3.3) in q.m. is given in the following.

THEOREM 3.1. *Denote $J = \{1 + j(n + 1)\}_{j=0}^{n-1} = \{1, n + 2, 2n + 3, \dots, n^2\}$ and let R be an $n^2 \times n^2$ diagonal matrix such that $r_{ii} = 1$ if $i \in J$ and $r_{ii} = 0$ if $i \notin J$. The randomly relaxed iteration* (3.3) *converges in q.m. if and only if*

$$\rho((I + p(A - I)) \otimes (I + p(A - I)) + p(1 - p)R((A - I) \otimes (A - I))) < 1. \tag{3.5}$$

Proof. The stochastic dynamic system $x(k+1) = \Psi(k)x(k)$ is asymptotically stable in q.m. if and only if $\rho(E[\Psi \otimes \Psi]) < 1$ (e.g., [7, p. 214]). Straightforward calculation gives

$$\begin{aligned} E[\Psi \otimes \Psi] &= E[(I + p(A-I) + Z(A-I)) \otimes (I + p(A-I) + Z(A-I)) \\ &= (I + p(A-I)) \otimes (I + p(A-I)) + E[(Z(A-I)) \otimes (Z(A-I))] \\ &= (I + p(A-I)) \otimes (I + p(A-I)) + E[Z \otimes Z]((A-I) \otimes (A-I)). \end{aligned}$$

Condition (3.5) follows when we note that $Z \otimes Z$ is a diagonal matrix with the diagonal blocks $\zeta_i Z$, so that $E[Z \otimes Z] = p(1-p)R$. □

The condition of Theorem 3.1 can also be expressed in terms of a matrix equation.

THEOREM 3.2. *Consider the matrix equation*

$$(3.6)\quad (I + p(A-I))'K(I + p(A-I)) - K + p(1-p)(A-I)'\bar{K}(A-I) = -G$$

where $\bar{K} = (k_{ii}\delta_{ij})$ (i.e., $\bar{K}$ is a diagonal matrix with its diagonal equal to the diagonal of K).

(i) *If the randomly relaxed iteration* (3.3) *converges in q.m., then equation* (3.6) *has a positive definite solution K for every positive definite matrix G.*

(ii) *If equation* (3.6) *has a positive definite solution K for some positive definite matrix G, then the randomly relaxed iteration* (3.3) *converges in q.m.*

Proof. We only need to show that $E[\Psi' K \Psi] - K$ equals the left side of (3.6), and the result will follow directly from Theorems 6.1 and 6.2 in [7, pp. 214–215]. Inserting $\Psi = I + p(A-I) + Z(A-I)$ we get

$$E[\Psi' K \Psi] = (I + p(A-I))'K(I + p(A-I)) + (A-I)'E[ZKZ](A-I).$$

Noting that $ZKZ = (\zeta_i \zeta_j k_{ij})$ so that $E[ZKZ] = p(1-p)\bar{K}$, we arrive at (3.6). □

It is interesting to note that with $p = 1$, Theorems 3.1 and 3.2 together reduce to the following well-known result. The Lyapunov matrix equation $A'KA - K = -G$ has a positive definite solution K for a positive definite G if and only if $\rho(A) < 1$.

Theorem 3.2 has an important consequence pertaining to almost sure convergence. Namely, if $E[\Psi' K \Psi] - K$ is negative definite for some positive definite K, then $V(x) = x'Kx$ is a suitable Lyapunov function and $V(x(k))$ is a positive supermartingale, showing the convergence of (3.3) in almost sure sense [5].

COROLLARY 3.3. *If the randomly relaxed iteration* (3.3) *converges in q.m., then it also converges almost surely; that is*

$$\Pr[\lim_{k \to \infty} \|x(k) - x^*\| = 0] = 1.$$

Because randomly relaxed iterations belong to the class of asynchronous iterations, one expects that the convergence condition (3.2) derived for asynchronous iterations is also sufficient for the convergence of (3.3) in q.m. This is indeed the case as is shown in the next theorem.

THEOREM 3.4. *Assume that $\rho(|A|) < 1$. Then the randomly relaxed iteration* (3.3) *converges in q.m. for all $p \in (0, 1]$.*

Proof. By a corollary to the Perron–Frobenius theorem [2, p. 150], $\rho(|A|) < 1$ implies that $\|A\|_w < 1$ for some w. From the equivalence of norms, $\|x\| \le c\|x\|_w$ for any x and for some constant c (precisely, $c = \sqrt{n}\cdot\max_i w_i$), so that

$$\begin{aligned} E\|x(k)\|^2 &\le c^2 E\|x(k)\|_w^2 = c^2 E\|\Psi(k-1)\cdots\Psi(0)x(0)\|_w^2 \\ &\le c^2 E[\|\Psi(k-1)\|_w^2 \cdots \|\Psi(0)\|_w^2 \|x(0)\|_w^2] = c^2\|x(0)\|_w^2 (E\|\Psi\|_w^2)^k, \end{aligned}$$

where the fact that $\Psi(k)$ are independent and identically distributed was used. Hence, $E\|x(k)\|^2 \to 0$ if $E\|\Psi\|_w^2 < 1$. Now the realizations of Ψ are matrices with the ith row equal to the ith row of A (denoted by a_i') or to the ith row of I. Hence, for any realization of Ψ and for any vector x we have

$$\begin{aligned} \|\Psi x\|_w &\le \max\{|x_1|/w_1, |a_1'x|/w_1, \ldots, |x_n|/w_n, |a_n'x|/w_n\} \\ &= \max\{\|x\|_w, \|Ax\|_w\} = \|x\|_w \end{aligned}$$

so that $\|\Psi\|_w \le 1$, and in particular, $\|\Psi\|_w < 1$ when $\Psi = A$. Likewise $\|\Psi\|_w^2 \le 1$ for all realizations and $\|\Psi\|_w^2 < 1$ for at least one realization of Ψ. Thus $E\|\Psi\|_w^2 < 1$ for all $p \in (0, 1]$. □

If $E\|x(k) - x^*\|^2 \to 0$, then also $E[x(k)] \to x^*$. Conversely, if the sequence $E[x(k)]$ diverges, then the randomly relaxed iteration cannot converge in q.m. We have the following result on the behavior of $E[x(k)]$.

THEOREM 3.5. *Let $x(k)$ be generated by the randomly relaxed iteration* (3.3). *Then $E[x(k)] \to x^*$ for all $x(0)$ if and only if*

$$\rho(I + p(A - I)) < 1. \tag{3.7}$$

Proof. $E[x(k)] = E[\Psi(k-1)\cdots\Psi(0)x(0)] = E[\Psi(k-1)]\cdots E[\Psi(0)]x(0) = E[\Psi]^k x(0) = (I + p(A - I))^k x(0)$, from which the result follows. □

The expected value convergence of (3.3) is thus equivalent to the convergence of an underrelaxed Jacobi iteration. It follows immediately from (3.7) that if A has real eigenvalues greater than unity, then the randomly relaxed iteration (3.3) is unstable. By the Perron–Frobenius theorem, $\rho(A)$ is an eigenvalue of A if A is nonnegative. Hence we have the following corollary to Theorems 3.4 and 3.5.

COROLLARY 3.6. *Let A be a nonnegative matrix. Then for any $p \in (0, 1]$, the randomly relaxed iteration* (3.3) *converges to x^* in q.m. if and only if $\rho(A) < 1$.*

This means that the convergence condition derived for totally asynchronous iterations, $\rho(|A|) < 1$, is tight for nonnegative matrices.

4. Remarks. Let us then consider some implications of the results presented. It follows from Theorem 3.1 that if the Jacobi iteration converges (i.e., $\rho(A) < 1$), then also the randomly relaxed iteration converges in q.m. for some $p < 1$. To see this, set $p = 1$ in (3.5) to arrive at $\rho(A \otimes A) < 1$. This is equivalent to $\rho(A) < 1$ so that, by the continuity of spectral radius, (3.5) must hold for some p close to 1. However, this does not guarantee that (3.5) is satisfied for all $p \in (0, 1)$. One can also find examples where condition (3.5) holds for some $p \in (0, 1)$ even though $\rho(A) > 1$ (cf. the numerical example in §5 below). This indicates that asynchronicity, as described by random relaxation, can in fact improve the stability of schemes such as the Jacobi iteration. An interesting research problem is whether a corresponding deterministic updating scheme could be uncovered that would have the same effect.

Theorem 3.2 can be used to derive other sufficient conditions for the convergence of randomly relaxed iterations. For example, inserting $K = I$ into (3.6) we get that $\rho(A'A) < 1$ guarantees q.m. convergence for all $p \in (0, 1]$. If A is normal ($AA' = A'A$), this is equivalent to $\rho(A) < 1$. This presents a considerable relaxation to condition (3.2).

5. Numerical example. Let us consider iterations with the matrix

$$A = \begin{bmatrix} -0.61 & -0.11 & -0.20 & 0.81 & 0.43 \\ 0.28 & 0.06 & 0.09 & -0.47 & -1.04 \\ -0.36 & -0.14 & 0.08 & 0.30 & -0.12 \\ 0.27 & 0.04 & -0.01 & -0.80 & -0.33 \\ 0.20 & -0.08 & 0.11 & -0.51 & -0.73 \end{bmatrix}.$$

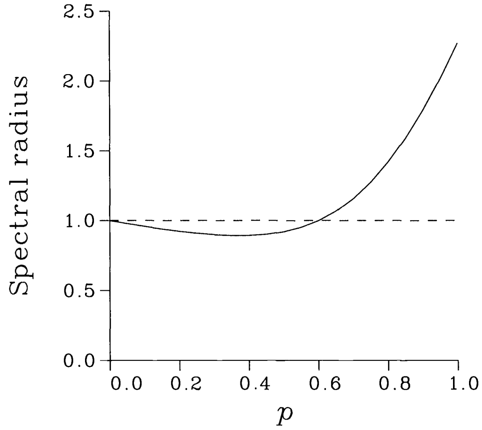


FIG. 5.1. *The spectral radius condition* (3.5) *of q.m. convergence as a function of p in the numerical example.*

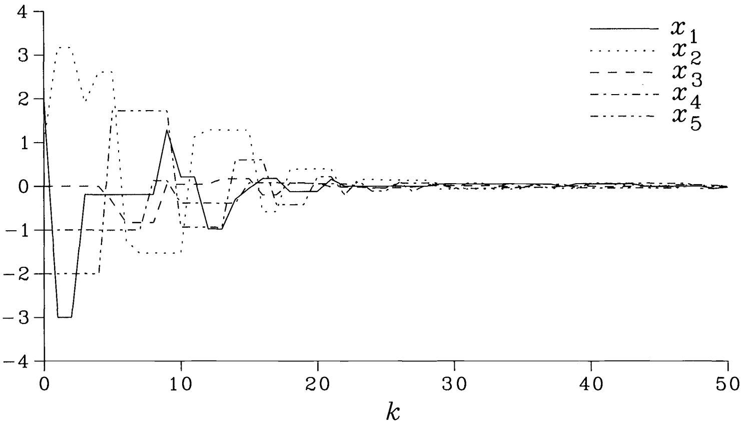


FIG. 5.2. *A simulation of the iteration* $x(k+1) = x(k) + \Xi(k)(A - I)x(k)$ *with* $p = 0.35$ *and* $x(0) = (2, 1, 0, -1, -2)$.

This provides an example where suitable asynchronism can be stabilizing. We have $\rho(|A|) \approx 1.6407$ and $\rho(A) \approx 1.5067$; in other words, the necessary condition for the convergence of asynchronous iterations is not satisfied and even the Jacobi iteration is unstable. Nevertheless, it turns out that condition (3.5) holds for all $p \in (0, \bar{p})$ where $\bar{p} \approx 0.60$, see Figure 5.1. When p decreases, the time intervals between coordinate updates become longer on the average. Hence, the interpretation is that randomly relaxed iterations converge in q.m. as well as almost surely if there is enough asynchronism. This phenomenon was first observed in numerical simulations of asynchronous Cournot processes in [12]. Figure 5.2 illustrates a simulation of the randomly relaxed iteration $x(k+1) = x(k) + \Xi(k)(A - I)x(k)$ with $p = 0.35$.

6. Conclusion. A stochastic model of zero-delay asynchronous iterations has been analyzed. Necessary and sufficient conditions for convergence in q.m. have been derived for linear systems. The results show that the convergence condition $\rho(|A|) < 1$ is tight for nonnegative matrices. It is also shown that asynchronous iterations, produced by random relaxation, can be stable even if the corresponding Jacobi iteration is unstable.

Acknowledgment. The author would like to thank Timo Eirola for helpful comments on the paper.

REFERENCES

[1] B. F. BEIDAS AND G. P. PAPAVASSILOPOULOS, *Convergence analysis of asynchronous linear iterations with stochastic delays*, Parallel Comput., 19 (1993), pp. 281–302.

[2] D. P. BERTSEKAS AND J. N. TSITSIKLIS, *Parallel and Distributed Computation*, Prentice-Hall, Englewood Cliffs, NJ, 1987.

[3] R. K. BRAYTON AND C. C. CONLEY, *Some results on the stability and instability of the backward differentiation methods with non-uniform time steps*, in Topics in Numerical Analysis, J. J. H. Miller, ed., Academic Press, London, 1973.

[4] R. K. BRAYTON AND C. H. TONG, *Stability of dynamical systems: A constructive approach*, IEEE Trans. Circuits Systems, 26 (1979), pp. 224–234.

[5] R. S. BUCY, *Stability and positive supermartingales*, J. Differential Equations, 1 (1965), pp. 151–155.

[6] D. CHAZAN AND W. MIRANKER, *Chaotic relaxation*, Linear Algebra Appl., 2 (1969), pp. 199–222.

[7] R. Z. HAS'MINSKIĬ, *Stochastic Stability of Differential Equations*, Sijthoff & Noordhoff, Alphen aan den Rijn, The Netherlands, 1980.

[8] R. P. LELAND, *Stability of asynchronous systems with Poisson transitions*, IEEE Trans. Automat. Control, 39 (1994), pp. 182–185.

[9] M. LOÈVE, *Probability Theory*, Van Nostrand, Toronto, 1955.

[10] G. P. PAPAVASSILOPOULOS, *Distributed algorithms with random processor failures*, IEEE Trans. Automat. Control, 39 (1994), pp. 1032–1036.

[11] J. N. TSITSIKLIS, *On the stability of asynchronous iterative processes*, Math. Systems Theory, 20 (1987), pp. 137–153.

[12] M. VERKAMA, R. P. HÄMÄLÄINEN, AND H. EHTAMO, *Multi-agent interaction processes: From oligopoly theory to decentralized artificial intelligence*, Group Decision and Negotiation, 2 (1992), pp. 137–159.

SIAM J. SCI. COMPUT.
Vol. 17, No. 4, pp. 913–919, July 1996

RAPID COMPUTATION OF THE DISCRETE FOURIER TRANSFORM*

CHRIS ANDERSON† AND MARIE DILLON DAHLEH‡

Abstract. Algorithms for the rapid computation of the forward and inverse discrete Fourier transform for points which are nonequispaced or whose number is unrestricted are presented. The computational procedure is based on approximation using a local Taylor series expansion and the fast Fourier transform (FFT). The forward transform for nonequispaced points is computed as the solution of a linear system involving the inverse Fourier transform. This latter system is solved using the iterative method GMRES with preconditioning. Numerical results are given to confirm the efficiency of the algorithms.

Key words. iterative methods, nonuniform fast Fourier transform

AMS subject classification. 65T20

1. Introduction.

The fast Fourier transform (FFT) [1] is a powerful tool used in numerous applications ranging from signal processing to rock mechanics [4]. In order to use the FFT one must have uniformly spaced data and numbers of points which are restricted to have certain values (e.g., a power of 2 or a product of primes). The goal of this paper is to describe a method for computing rapidly a forward and inverse discrete Fourier transform on sets of data points which are either nonequispaced and/or are unrestricted in number. Our method is similar in spirit to the papers of Dutt and Rokhlin [2], [3], in that the method relies on approximation of results obtained with the standard FFT. The major differences between the two approaches is that Dutt and Rokhlin utilize a fast multipole method to perform the approximation whereas we use a local Taylor series expansions. Our method also differs in the construction of the forward transform in that we employ a preconditioner and use an iterative method which does not require the transpose of the inverse transform matrix. We found our approach easier to implement because it is based on conventional computational techniques which are easy to program, namely the standard FFT and Taylor series expansions whose coefficients are computed spectrally.

In the next section we describe the specific computational problem associated with the discrete forward and inverse Fourier transform. In §3 we give a detailed description of our numerical algorithm. The key idea behind the inverse transform is to compute transformed values at a set of equispaced points using a standard FFT and then approximate the required values at the nonequispaced points (or arbitrary number) using local Taylor series expansions. The derivatives necessary for these expansions are computed spectrally. The use of local Taylor series expansions to represent the trigonometric sum is entirely appropriate since the sum is an analytic function. The computational procedure for the forward transform of a set of equispaced points of arbitrary number is the same as for the inverse transform (one just interchanges the notion of coefficients and function values). For nonequispaced points the forward transform is computed as the solution of a linear system of equations defined via the inverse transform. This system is solved iteratively using the generalized minimum residual method (GMRES) with preconditioning [5]. For each iteration an inverse Fourier transform must be computed, but, because the number of iterations is quite small, the method is efficient.

2. The problem.

In this paper we consider the forward Fourier transform as solving the trigonometric interpolation problem: Given the N distinct data points $\{x_j\}$ (not necessarily

*Received by the editors September 14, 1994; accepted for publication (in revised form) March 9, 1995.

†IBM T.J. Watson Research Center, Yorktown Heights, NY, 10598 (cra@watson.ibm.com). The work of this author was partially supported by ONR contract N00014-92-j-1890.

‡Department of Mathematics, UCLA, Los Angeles, CA 90024 (mdahleh@math.ucla.edu). The work of the second author was partially supported by the National Science Foundation grant DMS-9206192.

uniformly spaced) on the interval $[0, 2\pi]$ and associated function values $\{y_j\}$, find the N coefficients $\{\alpha_k\}$ so that the trigonometric sum formed with these coefficients interpolates the given data; i.e.,

$$y_j = \frac{1}{N}\sum_{k=0}^{N-1} \alpha_k e^{i\,k\,x_j} \tag{1}$$

for $j = 0, \dots, N-1$.

The set of values $\{\alpha_k\}$ are thus the coefficients of the data $\{y_j\}$ when expressed in the discrete basis obtained by evaluating the Fourier basis elements at the nodes $\{x_j\}$, i.e., the discrete forward Fourier transform of $\{y_j\}$.

If the points are equispaced, then the orthogonality of the discrete Fourier basis functions implies that

$$\alpha_k = \sum_{j=0}^{N-1} y_j e^{-i\,k\,x_j}. \tag{2}$$

Alternately, the inverse discrete Fourier transform consists of evaluating the trigonometric interpolant; i.e., given a set of Fourier coefficients $\{\beta_k\}$ and a set of points $\{x_j\}$ determine the values $\{y_j\}$ by evaluating the sums

$$y_j = \frac{1}{N}\sum_{k=0}^{N-1} \beta_k e^{i\,k\,x_j} \tag{3}$$

for $j = 0, \dots, N-1$.

3. The numerical method.

3.1. Inverse transform. Our procedure for the forward discrete Fourier transform uses our construction of the inverse discrete Fourier transform, and so we describe the inverse transform first. The procedure for the inverse transform consists of using the standard fast inverse transform on a set of $M = 2^q$ equispaced points to determine a set of local Taylor series expansions centered at these points. The Taylor series expansions are then evaluated at the desired set of nonequispaced points or equispaced points of arbitrary number.

Assume we have the N distinct points $\{x_j\}$ in an interval $[0, 2\pi]$ and we are given a set of N Fourier coefficients $\{\beta_k\}$. We first choose a value of M which is a power of 2, and for which $2\pi(N-1) \le M$. Let $\{\tilde{x}_m\}$ be the set of M equispaced points in the interval $[0, 2\pi]$. Then

- Construct a new set of coefficients $\{\tilde{\beta}_k\}$ by padding and scaling the coefficients $\{\beta_k\}$,

$$\tilde{\beta}_k = \begin{cases} \left(\frac{M}{N}\right)\beta_k, & k \le N, \\ 0, & N < k \le M. \end{cases}$$

  The scaling factor $\frac{M}{N}$ is introduced so that the inverse transform of the padded coefficients $\{\tilde{\beta}_k\}$ yields a function which is equivalent to that of the original coefficients at the points $\{x_j\}$.
- Using the discrete inverse FFT, compute the values and the first p derivatives of the function

$$\tilde{y}(x) = \frac{1}{M}\sum_{k=0}^{M-1} \tilde{\beta}_k e^{i\,k\,\tilde{x}} \tag{4}$$

at the points $\{\tilde{x}_m\}$. The derivatives are evaluated spectrally; i.e., the pth derivative is obtained by computing the inverse transform of a set of coefficients of the form $(i\,k)^p\,\tilde{\beta}_k$.

- For a given point x_j, determine the closest of the equispaced points, say $\tilde{x}^*$, to it. Using the values and derivatives of (4) at $\tilde{x}^*$ one approximates the value at the point x_j by using a pth order Taylor series expansion about $\tilde{x}^*$.

The order p of the Taylor series expansion used is determined by the desired precision required in the function values. If these values are required to a precision ϵ (which may be unit round-off) then the maximal order of the approximation required to do this is the value of p so that

$$\frac{\gamma}{(p+1)!} \leq \epsilon \tag{5}$$

where $\gamma = \max_{0\leq k\leq N-1} \mid \beta_k \mid$. This follows from the fact that the error in an order p Taylor expansion of an individual term in (4) with wavenumber k is bounded by

$$\frac{1}{M}\frac{\mid \tilde{\beta}_k \mid h^{p+1}k^{p+1}}{(p+1)!} = \frac{1}{N}\frac{\mid \beta_k \mid h^{p+1}k^{p+1}}{(p+1)!}.$$

Since $k \leq N-1$ and $h = \frac{2\pi}{M}$ we have

$$\frac{1}{N}\frac{\mid \beta_k \mid h^{p+1}k^{p+1}}{(p+1)!} \leq \frac{1}{N}\left(\frac{\mid \beta_k \mid}{(p+1)!}\right)\left(\frac{2\pi(N-1)}{M}\right)^{p+1} \leq \frac{1}{N}\frac{\mid \beta_k \mid}{(p+1)!}.$$

The error in approximating the sum of N Fourier components is thus bounded by N times the maximum of this value, i.e., (5). The estimate (5) is only a bound, and in the implementation of the approximation procedure one adaptively determines the size of p that one should use; i.e., one accumulates the Taylor series approximation term-by-term and stops when the error is within the desired precision. Our computational experiments indicate that far fewer terms than the number suggested by the error bound need be used.

3.2. Forward transform.

3.2.1. Equispaced points. If the points $\{x_j\}$ are equispaced then the forward transform of values $\{y_j\}$ is given by

$$\alpha_k = \sum_{j=0}^{N-1} y_j e^{-i\,k\,x_j}. \tag{6}$$

Since $x_j = j\left(\frac{2\pi}{N}\right)$, then we can rewrite the sum as

$$\alpha_k = \sum_{j=0}^{N-1} y_j e^{-i\,k\,x_j} = \sum_{j=0}^{N-1} y_j e^{-i\,jk\left(\frac{2\pi}{N}\right)} = \sum_{j=0}^{N-1} y_j e^{-i\,j\,z_k}.$$

Thus, the α_k's are the values of the function $\sum_{j=0}^{N-1} y_j e^{-i\,j\,z_k}$ evaluated at the N equispaced points $z_k = k\left(\frac{2\pi}{N}\right)$ in the interval $[0, 2\pi]$. The task of evaluating the forward transform for equispaced points thus has the same form as the evaluation of the inverse transform, so one can use the technique for the inverse transform described previously with only slight modifications. In particular, since the factor $\frac{1}{N}$ does not appear in the forward transform sum, one does not scale $\{y_j\}$ by $\left(\frac{M}{N}\right)$ when padding these values to create the M values $\{\tilde{y}_j\}$. Also, the error bound for the Taylor series approximation becomes

$$\frac{N\nu}{(p+1)!} \leq \epsilon \tag{7}$$

where $\nu = \max_{0 \leq j \leq N-1} \mid y_j \mid$. While this error bound does depend on N, the dependence is rather weak, and our computational experiments indicate that it is of no great concern.

The problem of the forward transform for nonequispaced points is more challenging because the the coefficients $\{\alpha_k\}$ are not given by the sum (6). Thus, a different procedure must be employed. In our procedure we build upon the fact that we can compute the inverse transform for nonequispaced points rapidly. In particular, the goal of the forward transform is to find coefficients $\{\alpha_k\}$ so that the inverse transform of these coefficients interpolates a given set of function values $\{y_j\}$. Expressed in matrix/vector notation, the forward transform consists of finding the vector of coefficients $\vec{\alpha}$ so that the linear system

$$A\vec{\alpha} = \vec{y} \tag{8}$$

is satisfied. A is the representation in matrix form of the inverse Fourier transform.

Our procedure for the forward transform is just to solve the linear system (8) for the coefficients $\vec{\alpha}$. We choose to solve this system iteratively, because this will involve operations of the form $A\vec{z}$ for vectors $\vec{z}$, i.e., applications of the inverse transform which can be computed efficiently using the technique described above.

The iterative method we choose to use was the GMRES method with preconditioning [5]. The GMRES method is a Krylov subspace iterative method for solving nonsymmetric linear systems. There are several advantages to this method. First, each iteration can be performed quickly because the bulk of the work is contained in the matrix multiplication step $A\vec{\alpha}^{(i)}$ which is just the evaluation of the inverse transform for each iterate. Second, it is optimal in the sense that it minimizes the residual in a given Krylov subspace. Third, it does not require the knowledge of A^T. The major disadvantage of this method is that it may require several iterations to converge. At each iteration, one computes the next basis element for the Krylov subspace. All previous basis elements are needed. If one needs a large number of iterations to converge, then one may need significant storage for this procedure. However, with the preconditioner described below, the method converges in a few iterations, and the storage requirements of GMRES does not pose a problem.

A preconditioner for the system (8) consists of an operator which takes a set of values $\vec{y}$ and returns another set of values $\vec{\alpha}$. After some experimentation we found that a satisfactory preconditioner was simply the forward Fourier transform of the values $\{y_j\}$ assuming that they are associated with equispaced points. (Thus the preconditioner is the exact inverse of the matrix A in the case of equispaced points.)

4. Numerical experiments. We have implemented our method in Fortran using double precision arithmetic. All the calculations were run on a Sparc 10. For the inverse transform we report two types of errors. The first is the relative ∞-norm which is defined to be

$$E_\infty = \frac{\max_{1 \leq i \leq N} |y_i^a - y_i|}{\max_{1 \leq i \leq N} |y_i|},$$

and the second is the relative 2-norm error defined by

$$E_2 = \left(\frac{\sum_{i=1}^{N} |y_i^a - y_i|^2}{\sum_{i=1}^{N} |y_i|^2} \right)^{\frac{1}{2}}$$

TABLE 1

p is the order of the Taylor series. Random .01 *perturbation for the uniform grid. Smooth coefficients. Error tolerance is* 10^{-10}.

N	p bound	p required	Sec.	Dir. sec.	E_∞	E_2
16	17	7	.01	0	1.1825×10^{-8}	1.2164×10^{-8}
32	18	7	.02	.01	1.1792×10^{-8}	1.5152×10^{-8}
64	18	7	.03	.03	8.2338×10^{-9}	1.2762×10^{-8}
128	18	7	.09	.12	5.8926×10^{-9}	1.3183×10^{-8}
256	18	8	.21	.49	4.8255×10^{-9}	1.4178×10^{-8}
512	19	8	.48	2.02	3.0118×10^{-9}	1.4329×10^{-8}
1024	19	8	1.02	8.32	2.4349×10^{-9}	1.4618×10^{-8}

TABLE 2

p is the order of the Taylor series. Random .1 *perturbation for the uniform grid. Smooth coefficients. Error tolerance is* 10^{-10}.

N	p bound	p required	Sec.	Dir. sec.	E_∞	E_2
16	17	11	.01	0	1.6951×10^{-8}	1.6017×10^{-8}
32	18	11	.03	.01	1.1192×10^{-8}	1.5587×10^{-8}
64	18	11	.05	.05	8.0280×10^{-9}	1.4234×10^{-8}
128	18	11	.12	.13	5.7089×10^{-9}	1.4554×10^{-8}
256	18	11	.3	.49	4.4565×10^{-9}	1.4754×10^{-8}
512	19	12	.68	2.02	3.8060×10^{-9}	1.4754×10^{-8}
1024	19	12	1.56	8.32	4.0123×10^{-9}	1.4547×10^{-8}

TABLE 3

p is the order of the Taylor series. Random perturbation of .1. *Random coefficients. Error tolerance is* 10^{-10}.

N	p bound	p required	Sec.	Dir. sec.	E_∞	E_2
16	17	11	.01	0	1.2223×10^{-8}	9.1421×10^{-9}
32	18	11	.02	.01	8.0229×10^{-9}	7.0160×10^{-9}
64	18	11	.05	.03	6.7398×10^{-9}	7.1751×10^{-9}
128	18	11	.12	.12	4.9839×10^{-9}	6.4089×10^{-9}
256	18	12	.32	.49	3.6305×10^{-9}	5.8583×10^{-9}
512	19	12	.74	2.16	3.4041×10^{-9}	4.8901×10^{-9}
1024	19	12	1.63	8.14	2.4524×10^{-9}	4.0691×10^{-9}

where y_i^a represents the approximation of the y_i. For the forward transform we report the final residual error.

4.1. Inverse transform experiments. The first set of experiments are concerned with the inverse transform. We consider two cases. In Tables 1–3, we present results for points which are not equispaced but whose number is a power of 2. We examine how the degree of nonuniformity of the points affects the number of terms in the Taylor series, p, needed to obtain a desired accuracy. We define the nonuniform grid as follows:

$$x_j = x_j^{\text{unif}} + h * \eta * factor$$

where x_j^{unif} represents the jth grid point on a uniform grid, h is the grid spacing, η a uniformly distributed random number between 0 and 1, and *factor* is either .01 (Table 1) or .1 (Table 2). As one can see from Tables 1 and 2, for both perturbation factors, our method is slower than the direct evaluation of (3) for $n < 64$ and faster for $n > 64$. One would expect as the perturbation factor increases from .01 to .1, the number of Taylor series terms required to

TABLE 4
p is the order of the Taylor series. Uniformly spaced nonpower of 2. Smooth coefficients. Error tolerance is 10^{-10}.

N	p bound	p required	Sec.	Dir. sec.	E_∞	E_2
11	17	8	.01	0	1.5206×10^{-8}	1.5441×10^{-8}
37	18	8	.04	.01	9.4308×10^{-9}	1.5486×10^{-8}
61	18	9	.04	.03	1.0360×10^{-8}	1.5632×10^{-8}
123	18	9	.1	.11	7.0404×10^{-9}	1.3590×10^{-8}
179	18	9	.26	.24	6.7775×10^{-9}	1.4953×10^{-8}
371	19	9	.49	1.04	5.2059×10^{-9}	1.4134×10^{-8}

TABLE 5
Forward transform. Random .01 perturbation from uniform grid.

N	Iterations	Sec.	Residual error
16	3	.12	5.8663×10^{-7}
32	3	.22	9.3676×10^{-7}
64	3	.46	8.8786×10^{-7}
128	3	.98	8.7670×10^{-7}
256	3	2.32	9.6058×10^{-7}
512	4	6.69	1.2012×10^{-8}

TABLE 6
Forward transform. Random .1 perturbation from uniform grid.

N	Iterations	Sec.	Residual error
16	8	.25	1.0326×10^{-7}
32	9	.59	4.09265×10^{-7}
64	9	1.27	9.6050×10^{-7}
128	10	2.85	3.8367×10^{-7}
256	10	6.95	7.1286×10^{-7}
512	10	15.5	6.5907×10^{-7}

obtain a given accuracy increases. This is in fact what happens. We use an error tolerance of 10^{-10}. The number of points for which our method is faster than the direct method is lower than the break-even point reported for a similar problem in either [2] or [3]. In [2], the break-even point occurs for $n \geq 256$ and in [3] the break-even point occurs for $n \geq 2048$.

For Tables 1 and 2, we use smooth Fourier coefficients generated by $\beta_k = \sin(x_k)$. In Table 3 we use randomly generated coefficients. The coefficients are contained in the interval $[0, 1]$. As one can see, for a given perturbation factor, the required number of Taylor series terms does not depend on the smoothness of the coefficients. Again, the order of the Taylor series required is substantially lower than the order predicted by our error estimates.

In Table 4 we present the results for a calculation with uniformly spaced points which are not a power of 2. One sees that our method is faster than the direct evaluation of (3) for calculations of more than 371 points.

4.2. Forward transform experiments. As we have mentioned previously, the computational task for the forward transform for equally spaced points has the same form as that for the inverse transform; therefore, there is no need to discuss it independently. Thus, in this section we only consider the case of the forward transform for data which is nonequispaced and whose number is a power of 2. In Tables 5 and 6, we consider a .01 perturbation and a .1 perturbation, respectively. For a given perturbation the number of GMRES iterations needed to converge remains almost constant. This is an indication that the preconditioner is doing a good job. Without the preconditioner the number of iterations increases dramatically with increasing N.

As the size of the perturbation increases, the number of iterations also increases. One should expect this behavior because the larger the perturbation the further the preconditioner is from the true matrix inverse. Even so, a ten-fold increase in the perturbation produces less than a three-fold increase in the CPU seconds needed for the calculation. In comparison with directly evaluating (1), the break-even point for the forward transform is about N = 4096. As better preconditioners are developed this number should drop.

In [2], Dutt and Rohklin present an iterative method for the solution of the algebraic problem. Our approach differs from theirs in that they use a conjugate gradient method whereas we use GMRES. Since the matrix A in (8) is not symmetric, one cannot use the conjugate gradient method directly for this system. Dutt and Rohklin therefore work with a reformulation of the matrix problem. By using GMRES we avoid using this reformulation. We also employ a preconditioner—a component we found to be critical for success.

5. Conclusions. In this paper we present a method for computing the forward and inverse discrete Fourier transform. The method for the inverse transform (and the forward transform of equispaced data with arbitrary numbers of points) is a combination of the standard FFT and approximation using local Taylor series expansions. Both the forward and inverse transforms are easy to implement and faster than directly evaluating (1) and (3) for a reasonably small number of points. The procedures reduce significantly the penalty of using the discrete Fourier transform on numbers of points which are not power of 2 or the product of primes.

The forward transform for nonequispaced data relies on the inverse transform and the iterative method GMRES with a preconditioner. The choice of the preconditioner was essential for the iterative method to converge rapidly. In our implementation we found that the procedure for the forward transform on nonequispaced points is less efficient than the direct method for numbers of points less than 4096. However, with improved coding and development of more appropriate preconditioners, this value should decrease. The methods presented here clearly extend to higher-dimensional discrete Fourier transforms.

REFERENCES

[1] J. W. Cooley and J. W. Tukey, *An algorithm for the machine computation of complex Fourier series*, Math. Comp., 19 (1965), pp. 297–301.

[2] A. Dutt and V. Rokhlin, *Fast Fourier transforms for nonequispaced data*, SIAM J. Sci. Comput. Statist., 14 (1993), pp. 1368–1393.

[3] ———, *Fast Fourier Transforms for Nonequispaced Data* II, Research Report Yaleu/dcs/rr-980, Yale University, New Haven, CT, August 1993.

[4] A. P. Peirce, S. Spottiswoode, and J. A. L. Napier, *The spectral boundary element method: A new window on boundary elements in rock mechanics*, Internat. J. Rock Mech. Min. Sci. Geomech. Abstr., 29 (1992), pp. 379–400.

[5] Y. Saad and M. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

SIAM J. SCI. COMPUT.
Vol. 17, No. 4, pp. 920–941, July 1996

009

FAST RECURSIVE LEAST SQUARES ADAPTIVE FILTERING BY FAST FOURIER TRANSFORM-BASED CONJUGATE GRADIENT ITERATIONS*

MICHAEL K. NG† AND ROBERT J. PLEMMONS‡

Abstract. Recursive least squares (RLS) estimations are used extensively in many signal processing and control applications. In this paper we consider RLS with sliding data windows involving multiple (rank k) updating and downdating computations. The least squares estimator can be found by solving a near-Toeplitz matrix system at each step. Our approach is to employ the preconditioned conjugate gradient method with circulant preconditioners to solve such systems. Here we iterate in the time domain (using Toeplitz matrix-vector multiplications) and precondition in the Fourier domain, so that the fast Fourier transform (FFT) is used throughout the computations. The circulant preconditioners are derived from the spectral properties of the given input stochastic process. When the input stochastic process is stationary, we prove that with probability 1, the spectrum of the preconditioned system is clustered around 1 and the method converges superlinearly provided that a sufficient number of data samples are taken, i.e., the length of the sliding window is sufficiently long. In the case of point-processing ($k = 1$), our method requires $O(n \log n)$ operations per adaptive filter input where n is the number of least squares estimators. In the case of block-processing ($k \geq n$), our method requires only $O(\log n)$ operations per adaptive filter input. A simple method is given for tracking the spectral condition number of the data matrix at each step, and numerical experiments are reported in order to illustrate the effectiveness of our FFT-based method for fast RLS filtering.

Key words. signal processing, adaptive filter, recursive least squares, Toeplitz matrix, circulant matrix, covariance matrix, fast Fourier transform, preconditioned conjugate gradient method, sliding windows, condition estimation

AMS subject classifications. 65F10, 65F15, 43E10

1. Introduction.

1.1. Background.

Adaptive filters are used extensively in many signal processing and control applications: for instance, in system identification, equalization of telephone channels, spectrum analysis, noise cancellation, echo cancellation, and linear predictive coding. A large number of books and papers have been written on various aspects of these applications, notably [13, pp. 17–75], [2, pp. 1–5], and [24]. In these applications the *recursive least squares* (RLS) algorithm is a well-known and extremely powerful tool.

The standard *linear least squares problem* can be posed as follows: given a complex M-by-n data matrix X with full column rank n (so that X^*X is Hermitian positive definite) and an M-vector $\mathbf{d}$ (desired signal vector), find the n-vector $\mathbf{w}$ (filter coefficient vector) that solves

$$\min \|\mathbf{d} - X\mathbf{w}\|_2, \tag{1.1}$$

where $\|\cdot\|_2$ denotes the usual Euclidean norm. Here X^* denotes the conjugate transpose. The solution to (1.1) is given by

$$\mathbf{w} = (X^*X)^{-1}X^*\mathbf{d}. \tag{1.2}$$

Here X^*X is often called the normal equations matrix [11, p. 142], and is known as the information matrix for (1.1) in the signal processing literature [13, p. 383]. It measures the information content in the experiment leading to (1.1).

*Received by the editors April 12, 1993; accepted for publication (in revised form) February 27, 1995.

†Department of Mathematics, The Chinese University of Hong Kong, Shatin, Hong Kong. Current address: Computer Sciences Laboratory, The Australian National University, Canbarra, ACT 0200, Australia.

‡Computer Science Department, Wake Forest University, Box 7388, Winston-Salem, NC 27109. This research was supported by the Air Force Office of Scientific Research under grant F49620–94–1–0261, and the National Science Foundation under grant CCR–92–01105.

In RLS computations arising in adaptive control and signal processing, it is required that one recalculate $\mathbf{w}$ when observations (i.e., equations) are successively added to, or deleted from, the problem (1.1); i.e., the least squares estimator at step t is found by solving for the n-vector $\mathbf{w}(t)$ in

$$\min ||\mathbf{d}(t) - X(t)\mathbf{w}(t)||_2. \tag{1.3}$$

Here $\mathbf{d}(t)$ is a desired signal M-vector at step t, and $X(t)$ is a corresponding M-by-n data matrix at step t. In the area of signal processing, RLS algorithms are often used to process signals that result from time-varying environments. RLS estimations for a time-varying finite impulse response or a transversal filter are often obtained by limiting the filter memory. The most common technique uses an exponential data weighting infinite memory method controlled by a forgetting factor γ, with $0 < \gamma \leq 1$ [13, p. 478]. This facilitates simplified computations in contrast to finite memory sliding data window methods, but at the cost of diminished tracking and stability characteristics [15, 23]. In many applications it is desirable to use a true finite memory algorithm, i.e., a sliding window algorithm, in order to avoid undesired effects from data in the distant past. A typical situation is where the parameters of the underlying model that generates the signal are subject to jump-type variations of random amplitudes [26, 27]. For a sliding window M-by-n data matrix, the least squares estimators can be computed by modifying the Cholesky factor of the normal equations with $O(n^2)$ operations per adaptive filter input where n is number of filter coefficients; see, for instance, [21].

Recently, fast recursive least squares (FRLS) algorithms have been a topic of considerable interest because of their low computational cost [13, pp. 583–584]. In most applications in signal processing, for instance linear predictive coding and system identification, the Toeplitz (displacement) structure of the data matrix allows one to develop computationally efficient algorithms, which are fast in the sense that they require only $O(n)$ operations per adaptive filter input. But the numerical stability of these FRLS algorithms has always been in question [14, 15, 23]. In particular, Luo and Qiao and Qiao [15, 23] have recently shown that *all known infinite memory FRLS algorithms are unstable* when the forgetting factor, used to diminish the effects of the old data, is less than 1.

The purpose of this paper is to propose a new FFT-based iterative RLS algorithm with reasonable complexity for computing least squares estimators recursively, that may also avoid some of the instability problems associated with direct FRLS methods.

1.2. FFT-based preconditioned iterative method. The use of conjugate gradient methods with circulant preconditioners for solving n-by-n Toeplitz systems $A_n\mathbf{u} = \mathbf{v}$ has been studied extensively in recent years. The key idea is to use n-by-n circulant matrices S_n to precondition the Toeplitz systems so as to speed up the convergence rate of the method (see, e.g., Chan and Strang [4]). That means, instead of solving the original Toeplitz system, one solves the preconditioned system

$$S_n^{-1}A_n\mathbf{u} = S_n^{-1}\mathbf{v}$$

by the conjugate gradient method.

Circulant matrices can always be diagonalized by the discrete Fourier matrix F_n with entries given by $[F_n]_{j,k} = (1/\sqrt{n})e^{-2\pi ijk/n}$. Thus linear systems with circulants can be solved in $O(n \log n)$ operations using the fast Fourier transform (FFT). Also, matrix-vector multiplications $A_n\mathbf{v}$ can be computed using the FFT in $O(n \log n)$ operations, by first embedding A_n into a $2n$-by-$2n$ circulant matrix; see [4]. It follows that the number of operations per iteration of the preconditioned conjugate gradient (PCG) method is of order $O(n \log n)$, using the FFT.

The convergence rate of this FFT-based PCG method has been analyzed by Chan and Strang [4]. They proved that if the diagonals of the Toeplitz matrix A_n are Fourier coefficients of a positive function in the Wiener class, then the spectrum of the preconditioned system $S_n^{-1}A_n$ will be clustered around 1 for large n, and thus the method will converge superlinearly. More precisely, for all $\epsilon > 0$, there exists a constant $c(\epsilon) > 0$ such that the error vector $\mathbf{b}_j$ of the preconditioned conjugate gradient method at the jth iteration satisfies

$$||\mathbf{b}_j||_{S_n^{-1/2}A_nS_n^{-1/2}} \leq c(\epsilon)\epsilon^j||\mathbf{b}_0||_{S_n^{-1/2}A_nS_n^{-1/2}}$$

when n is sufficiently large. Here

$$||\mathbf{v}||^2_{S_n^{-1/2}A_nS_n^{-1/2}} = \mathbf{v}^*S_n^{-1/2}A_nS_n^{-1/2}\mathbf{v}.$$

Hence the complexity of solving a large class of Toeplitz systems can be reduced to $O(n \log n)$ operations in such situations.

We remark that circulant approximations to Toeplitz matrices have been considered and used for some time in image processing, signal processing and time series analysis; see [8, 18] for references. Besides Strang's circulant preconditioner S_n, several other successful circulant preconditioners have been proposed and analyzed; see [6, 9, 25]. Recently, the use of circulant preconditioners for Toeplitz least squares problems was considered by Chan, Nagy and Plemmons [7, 8] and Ng and Chan [18]. In these papers, formal convergence results are established for the least squares problems, and some applications to signal and image processing are derived.

1.3. Outline. In this paper, we consider the recursive least squares computations where the data matrices are assumed to have a Toeplitz (displacement) structure. We propose a new algorithm for computing least squares estimators recursively. Our approach uses sliding data windows involving multiple updating and downdating computations, for superior tracking capabilities. When $X(t)$ is an M-by-n rectangular data matrix (M is the length of the sliding window) with full column rank, then the least squares estimator $\mathbf{w}(t)$ at step t can be obtained by solving the normal equations

$$X(t)^*X(t)\mathbf{w}(t) = X(t)^*\mathbf{d}(t). \tag{1.4}$$

We note that $X(t)^*X(t)$, although generally not Toeplitz, is an n-by-n "near-Toeplitz" matrix, as it can be written in the form $T(t) - L(t)^*L(t) - U(t)^*U(t)$, where $T(t)$ is Toeplitz, and $L(t)$ and $U(t)$ are lower triangular and upper triangular Toeplitz matrices, respectively (see §2).

Our approach is to apply the preconditioned conjugate gradient algorithm with circulant preconditioners to solve the system (1.4) at each step t. In our algorithm, the proposed n-by-n circulant preconditioner $C(t)$ is taken to be an approximation to $T(t)$. We prove that if the input stochastic process is stationary and its underlying spectral density function is positive and in the Wiener class, then our circulant preconditioner $C(t)$ will be positive definite, and its smallest eigenvalue will be uniformly bounded away from zero with probability 1, provided that sufficiently large numbers of data samples are taken, i.e., M is sufficiently large. Under the same assumptions, we also prove that the spectrum of the preconditioned matrix $C(t)^{-1}X(t)^*X(t)$ is clustered around 1 with probability 1. Thus, when we apply the conjugate gradient method to the preconditioned system, the method converges superlinearly with probability 1.

We also derive a convergence analysis of the sliding window RLS involving multiple updating and downdating computations. We prove that with probability 1, if the desired response and the input stationary stochastic process are related by a *multiple linear regression model*, then the ℓ_2 norm of the difference between the least squares estimator computed by the sliding window RLS algorithm and the constant regression parameter is bounded by a constant, provided that a large number of data samples is taken, i.e., M is sufficiently large. Under the same assumptions, we prove that with probability 1, the average least squares error of the sliding window RLS is also bounded by a constant.

The outline of the paper is as follows. In §2, we first formulate the sliding window RLS method for a Toeplitz data matrix and analyze the convergence rate of the preconditioned conjugate gradient method probabilistically. In §3, we present our FFT-based RLS algorithms and study the convergence of the sliding window RLS algorithm. In §4, numerical experiments are reported for the sliding window RLS scheme in order to illustrate the effectiveness of the method. A simple method is also given for tracking the spectral condition number of the data matrix at each step. Some concluding remarks are given in §5.

2. Sliding window RLS with Toeplitz data matrix. To present the sliding window RLS method, we first introduce some notation from adaptive filter theory [13, p. 18].

- discrete time index or step: t,
- order of filter: n,
- input sample scalar at time t: $x(t)$,
- input sample row vector at time t: $\mathbf{x}(t)^* = [\overline{x(t)}, \overline{x(t-1)}, \ldots, \overline{x(t-n+1)}]$,
- optimal filter coefficient column vector (the least squares estimator) at time t: $\mathbf{w}(t) = [w_1(t), w_2(t), \ldots, w_n(t)]^T$,
- desired signal scalar at time t: $d(t)$,
- filter output scalar at time t: $o(t) = \mathbf{x}(t)^*\mathbf{w}(t)$,
- difference (estimation error) between the desired response $d(n)$ and the output $o(t)$ produced by the filter at time t: $e(t) = d(t) - o(t)$,
- length of sliding window: M (we assume that $M \geq n$),
- rank number for updating and downdating computations: k ($k \leq M$).

In the sliding window RLS method, the corresponding data matrix at step t ($t \geq M$) is an M-by-n rectangular Toeplitz matrix of the form

$$X(t) = \begin{bmatrix} \overline{x(t-M+1)} & \cdots & \cdots & \overline{x(t-M-n+2)} \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ \vdots & \ddots & & \overline{x(t-M+1)} \\ \vdots & & \ddots & \vdots \\ \overline{x(t)} & \cdots & \cdots & \overline{x(t-n+1)} \end{bmatrix}. \tag{2.1}$$

We assume that $t - M - n + 2 \geq 1$ in $X(t)$. Thus all data samples are available at step t except when $t = M$. In this case, we let $x(j) = 0$ for $j \leq 0$. Moreover, we always assume that the data matrix $X(t)$ is of full column rank n at each updating and downdating step $t = M, M+k, M+2k, \ldots$. By minimizing the estimation error over the steps from $t - M + 1$ to t,

$$\sum_{j=t-M+1}^{t} |e(j)|^2, \tag{2.2}$$

the least squares estimator $\mathbf{w}(t)$ can be found by solving the least squares problem

$$\min ||\mathbf{d}(t) - X(t)\mathbf{w}(t)||_2.$$

Here $\mathbf{d}(t)$ is a known M-vector given by

$$\mathbf{d}(t) = [d(t-M+1), d(t-M+2), \ldots, d(t)]^T. \tag{2.3}$$

It follows that the solution $\mathbf{w}(t)$ can be obtained by solving normal equations

$$X(t)^*X(t)\mathbf{w}(t) = X(t)^*\mathbf{d}(t).$$

Since the data matrix $X(t)$ is assumed to have full column rank n, the normal equations matrix is nonsingular. We remark that the data matrix has a special structure. Each row of $X(t)$ is a right-shifted version of the previous row. By utilizing this special rectangular Toeplitz (displacement) structure of the data matrix, the normal equations matrix $X^*(t)X(t)$ can be written in the form

$$X(t)^*X(t) = T(t) - L(t)^*L(t) - U(t)^*U(t), \tag{2.4}$$

where $T(t)$ is Hermitian and Toeplitz, and $L(t)$ and $U(t)$ are lower triangular and upper triangular Toeplitz matrices, respectively (see, e.g., [18]). As the product of a lower triangular Toeplitz matrix and an upper triangular Toeplitz matrix is not Toeplitz, in general, the normal equations matrix $X(t)^*X(t)$ is not Toeplitz. We call $X(t)^*X(t)$ *near-Toeplitz.*

Here, the first column of the Hermitian Toeplitz matrix $T(t)$ in (2.4) is given by

$$[\gamma_0(t), \gamma_1(t), \ldots, \gamma_{n-1}(t)],$$

where

$$\gamma_j(t) = \sum_{s=t-M-n+2}^{t-|j|} x(s)\overline{x(s+|j|)}, \quad j = 0, 1, \ldots, n-1.$$

Moreover, in the statistics literature, if the input stochastic process is stationary, the parameters

$$\hat{r}_j(t) \equiv \frac{\gamma_j(t)}{M+n-1}, \quad j = 0, 1, \ldots, n-1, \tag{2.5}$$

are called estimators of the autocovariances r_j of the stationary process; see, for instance, Priestley [22, p. 322]. We also note that the first column of the lower triangular Toeplitz matrix $L(t)$ and the first row of the upper triangular Toeplitz matrix $U(t)$ are given by

$$[0, \overline{x(t-M-n+2)}, \ldots, \overline{x(t-M)}]^T \quad \text{and} \quad [0, \overline{x(t)}, \ldots, \overline{x(t-n+2)}],$$

respectively (see, e.g., [18]).

2.1. Updating and downdating computations. Figure 2.1 displays the sliding data window RLS method. Here, k rows, denoted by the k-by-n updating matrix $Y(t)$, are added, and k rows, denoted by the k-by-n downdating matrix $Z(t)$, are removed, forming the M-by-n data matrix $X(t+k)$ at step $t+k$. We note that the k-by-n updating matrix $Y(t)$ and downdating matrix $Z(t)$ are given by

$$Y(t) = \begin{bmatrix} \overline{x(t+1)} & \cdots & \cdots & \cdots & \overline{x(t-n+2)} \\ \vdots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ \overline{x(t+k)} & \cdots & \cdots & \cdots & \overline{x(t+k-n+1)} \end{bmatrix}$$

and

$$Z(t) = \begin{bmatrix} \overline{x(t-M+1)} & \cdots & \cdots & \cdots & \overline{x(t-M-n+2)} \\ \vdots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ \overline{x(t-M+k)} & \cdots & \cdots & \cdots & \overline{x(t-M+k-n+1)} \end{bmatrix}.$$

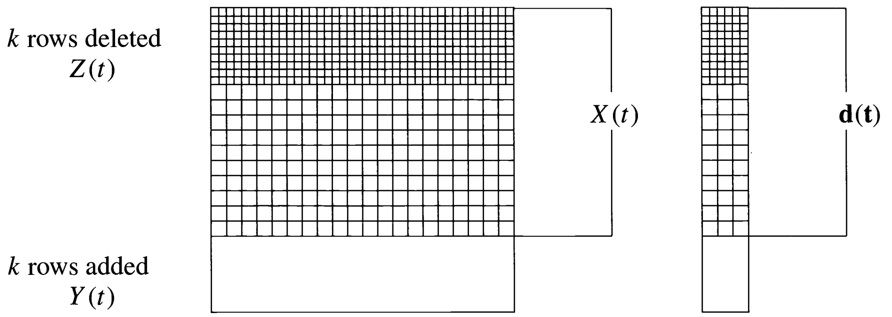


FIG. 2.1. *Sliding window RLS method: Updating with $Y(t)$ and downdating with $Z(t)$.*

The right-hand side data vector $\mathbf{d}(t+k)$ is modified in a corresponding fashion. In practice, the sliding window length M can be modified adaptively in the recursive computations. This is accomplished easily. For instance, to increase (decrease) M one can downdate with fewer (more) rows than used for updating. For simplicity of notation, however, we describe only the case where the window length M is fixed, i.e., we describe the FFT-based sliding window RLS algorithm only for combined rank k updates and downdates. Next we show how to generate the normal equations matrix $X(t+k)^*X(t+k)$ and the right-hand side vector of the normal equations $X(t+k)^*\mathbf{d}(t+k)$ at the next step.

We use the fact that at the step $t+k$, the normal equations matrix can be written as the following form which corresponds to (2.4),

$$X(t+k)^*X(t+k) = T(t+k) - L(t+k)^*L(t+k) - U(t+k)^*U(t+k),$$

where the first column of $T(t+k)$ is given by

$$[\gamma_0(t+k), \gamma_1(t+k), \dots, \gamma_{n-1}(t+k)]$$

and

$$\gamma_j(t+k) = \sum_{s=t-M+k-n+2}^{t+k-|j|} x(s)\overline{x(s+|j|)}.$$

Here the first column of the lower triangular and the first row of the upper triangular Toeplitz matrices $L(t+k)$ and $U(t+k)$ are given by

$$[0, \overline{x(t-M+k-n+2)}, \dots, \overline{x(t-M+k)}]^T \quad \text{and} \quad [0, \overline{x(t+k)}, \dots, \overline{x(t+k-n+2)}]$$

respectively. Both the matrices $L(t+k)$ and $U(t+k)$ can be easily generated from the updating matrix $Y(t)$ and downdating matrix $Z(t)$. Moreover, the first column of the Toeplitz matrix $T(t+k)$ can be generated by

$$T(t+k)\mathbf{e_1} = T(t)\mathbf{e_1} + Y(t)^*Y(t)\mathbf{e_1} - HZ(t)^T(Z(t)^T)^*\mathbf{e_n},$$

where H is the n-by-n antidiagonal identity matrix, and $\mathbf{e_1}$ and $\mathbf{e_n}$ are the first unit and last unit vectors. By utilizing the special Toeplitz (displacement) structure of the data matrices $Y(t)$ and $Z(t)$, the first column of $T(t+k)$ can be obtained by the FFT in $O(\max\{n,k\}\log n)$

operations. We note that the right-hand side vector $X(t+k)^*\mathbf{d}(t+k)$ of the normal equations at the step $t+k$ can be obtained in $O(\max\{k,n\}\log n)$ operations by using similar updating and downdating computations, i.e.,

$$X(t+k)^*\mathbf{d}(t+k) = X(t)^*\mathbf{d}(t) - Z^*(t)\mathbf{d}_d(t) + Y^*(t)\mathbf{d}_u(t),$$

where the vectors $\mathbf{d}_d(t)$ and $\mathbf{d}_u(t)$ are given by

$$\mathbf{d}_d(t) = [d(t-M+1), d(t-M+2), \ldots, d(t-M+k)]^T$$

and

$$\mathbf{d}_u(t) = [d(t+1), d(t+2), \ldots, d(t+k)]^T,$$

respectively.

We remark that in the case of point-processing, i.e., when $k = 1$, the updating and downdating computations for the sliding window RLS process is much simpler than the case of block-processing ($k > 1$). More precisely, we do not need to use the FFT to compute the first column of the Toeplitz matrix $T(t+1)$ and the right-hand side vector at the step $t+1$. The first column of $T(t+1)$ can be generated by

$$T(t+1)\mathbf{e}_1 = T(t)\mathbf{e}_1 + \overline{x(t+1)}\mathbf{x}(t) - x(t-M-n+2)H\mathbf{x}(t-M+1).$$

Similarly, the right-hand side vector $X^*(t+1)\mathbf{d}(t+1)$ is computed by the following formula:

$$X^*(t+1)\mathbf{d}(t+1) = X(t)^*\mathbf{d}(t) - d(t-M+1)\mathbf{x}(t-M+1) + d(t+1)\mathbf{x}(t+1).$$

Here, the complexity of these rank 1 updating and downdating computations is $O(n)$.

Recall that our propose is to use the circulant preconditioned conjugate gradient method to compute the filter coefficient vectors at each adaptive time step. In the next section, we introduce our circulant preconditioner and show that the circulant preconditioned system converges very quickly at each adaptive time step.

2.2. FFT-based preconditioned conjugate gradient iterations. In this paper, we only focus on the optimal circulant preconditioner $C(t)$ at the step t, which is defined to be the minimizer of $||Q_n - T(t)||_F$ over all n-by-n circulant matrices Q_n [9]. Here $||\cdot||_F$ denotes the Frobenius norm. The following lemma gives the spectral properties of the optimal circulant preconditioner. Its proof can be found in Tyrtyshnikov [25].

LEMMA 2.1 (of Tyrtyshnikov). *Let A_n be an arbitrary n-by-n Hermitian matrix and C_n be the optimal preconditioner, which is defined to be the minimizer of $||Q_n - A_n||_F$ over all n-by-n circulant matrices Q_n. Then C_n is Hermitian and*

$$\lambda_{\min}(A_n) \le \lambda_{\min}(C_n) \le \lambda_{\max}(C_n) \le \lambda_{\max}(A_n), \tag{2.6}$$

where $\lambda_{\max}(\cdot)$ and $\lambda_{\min}(\cdot)$ denote the largest and the smallest eigenvalues, respectively. In particular, if A_n is positive definite, then C_n is also positive definite.

We remark that most of the other circulant preconditioners do not satisfy (2.6); see Chan and Yeung [6]. Since $T(t)$ is a Toeplitz matrix, the first column of $C(t)$ can be obtained in $O(n\log n)$ operations by the FFT; i.e., the circulant matrix $C(t)$ can be generated in $O(n\log n)$ operations [9].

Next we analyze the convergence rate of the conjugate gradient method when applied to solving the preconditioned system, which can be expressed as

$$C(t)^{-1}[X(t)^*X(t)]\mathbf{w}(t) = C(t)^{-1}X(t)^*\mathbf{d}(t),$$

at each step t. As we deal with data samples from random input processes, the convergence rate will be considered in a probabilistic way, which is different from the deterministic case discussed in §1.2. For simplified notation in the following discussions, we define an integer

$$J = M + n - 1,$$

which refers to *number of data samples* given in the data matrix $X(t)$ (cf. (2.1)).

We first make the following practical assumption (A) on the input signal process, so that results of the convergence rate can be derived.

(A1) The input discrete-time stochastic process is stationary.

(A2) The underlying spectral density function $f(\theta)$ (see Haykin [13, pp. 116–117] for definition) of the input process is real-valued, positive, and in the Wiener class; i.e., the autocovariances $\{r_k\}$ of the process are absolutely summable:

$$\sum_{k=-\infty}^{\infty} |r_k| \leq \alpha < \infty. \tag{2.7}$$

(A3) The variances of the estimators $\hat{r}_k(t)$ given in (2.5) are bounded by

$$\mathrm{Var}(\hat{r}_k(t)) \equiv \mathrm{Var}\left(\frac{\gamma_k(t)}{J}\right) \leq \frac{\beta}{J}, \quad k = 0, \pm 1, \pm 2, \dots, \tag{2.8}$$

where β is a constant.

(A4) The stationary process has zero-mean, i.e., $\mathcal{E}(x(t)) = \mu = 0$ for all t where $\mathcal{E}$ is the expectation operator.

Here are some remarks on the assumptions.

1. The assumption (A1) is often true in signal-processing applications. For instance (A1) holds for *autoregressive* (AR) *processes*, *moving-average* (MA) *processes* and *autoregressive and moving-average* (ARMA) *processes*, which are commonly used as input stochastic stationary processes, see Haykin [13, pp. 90–94].
2. In time-series analysis, assumption (A2) is often valid. For example, the spectral density functions of ARMA processes are rational functions [3, p. 121]. The positiveness of the spectral density function can be guaranteed by the causality of the process [3, p. 85], whereas the absolute summability of the autocovariances can be assured by the invertibility of the process [3, p. 86]. In the time-series literature, the Hermitian Toeplitz matrix R_n with its first column given by

$$[r_0, r_1, \dots, r_{n-2}, r_{n-1}]^T$$

is called the covariance matrix of the input stochastic stationary process [13, p. 82]. Moreover, we have the following lemma about the spectrum $\sigma(R_n)$ of R_n given in Haykin [13, p. 139].

LEMMA 2.2. *Let the input process satisfy assumptions* (A1) *and* (A2). *Then the spectrum $\sigma(R_n)$ of R_n satisfies*

$$\sigma(R_n) \subseteq [f_{\min}, f_{\max}], \quad \forall n \geq 1, \tag{2.9}$$

where $f_{\min}$ and $f_{\max}$ are the minimum and maximum values of $f(\theta)$, respectively. In particular, if $f(\theta)$ is positive, then R_n is positive definite.

We note that further remarks on the assumptions can also be found in [18]. We first establish the result that the smallest eigenvalue and the largest eigenvalue of $X(t)^*X(t)$ are uniformly bounded away from zero with probability 1. Basically, the lemma states that the normal equations matrix is a good approximation to the covariance matrix R_n in the stochastic sense. In time-series analysis and signal processing, the normal equations matrix is sometimes called a sample covariance matrix [22, pp. 321–322], [13, pp. 378–379].

LEMMA 2.3 (Ng and Chan [18, Thm. 2 and Lem. 4]). *Let the input process satisfy assumption* (A). *Then for any given* $\epsilon > 0$ *and* $0 < \delta < 1$, *there exists a positive integer* N *such that for* $n > N$,

$$\Pr\left\{\lambda_{\min}\left(\frac{1}{J}[X(t)^*X(t)]\right) \geq \lambda_{\min}(R_n) - \epsilon \geq f_{\min} - \epsilon\right\} > 1 - \delta$$

and

$$\Pr\left\{\lambda_{\max}\left(\frac{1}{J}[X(t)^*X(t)]\right) \leq \lambda_{\max}(R_n) + \epsilon \leq f_{\max} + \epsilon\right\} > 1 - \delta,$$

provided that the number of data samples $J (= M + n - 1)$ *is sufficiently large* $(J \gg n)$.

Next we prove that the circulant preconditioned matrices have clustered spectra. The result is stated as the following theorem.

THEOREM 2.4. *Let the input process satisfy assumption* (A). *Then, for any given* $\epsilon > 0$ *and* $0 < \delta < 1$, *there exist positive integers* K *and* N *such that for* $n > N$,

$$\Pr\{\textit{at most } K \textit{ eigenvalues of } I_n - C(t)^{-1}X(t)^*X(t) \textit{ have absolute value} > \epsilon\} > 1 - \delta,$$

provided that the number of data samples J *is sufficiently large* $(J \gg n)$.

Proof. We define the following events:

$E_1 = \{$at most K eigenvalues of $\frac{1}{J}[C(t) - X(t)^*X(t)]$ have absolute value $> \epsilon\}$,
$E_2 = \{\lambda_{\min}(C(t))$ is uniformly bounded away from zero$\}$, and
$E_3 = \{$at most K eigenvalues of $I_n - C(t)^{-1}X(t)^*X(t)$ have absolute value $> \epsilon\}$.

By the results proved in Ng and Chan [18, Thm. 3 and Lem. 4], we have, for any given $\epsilon > 0$ and $0 < \delta < 1$, that there exist positive integers K and N such that for $n > N$, Pr $\{E_1\} > 1 - \delta$. Using Lemmas 2.2 and 2.3, we have Pr $\{E_2\} > 1 - \delta$. Then we note that

$$\Pr\{E_1 \text{ and } E_2\} = \Pr\{E_1\} + \Pr\{E_2\} - \Pr\{E_1 \text{ or } E_2\} \geq 1 - 2\delta.$$

Since events E_1 and E_2 together imply E_3, the theorem follows. □

Using Theorem 2.4, we can easily show that the conjugate gradient method, when applied to the preconditioned matrix $C(t)^{-1}X(t)^*X(t)$, converges superlinearly with probability 1 provided that the number of data samples $J(= M + n - 1)$ is sufficiently large, i.e., M is sufficiently large. For details of the proof of the superlinearly convergence rate, see Chan and Strang [4]. Thus the number of iterations required to achieve a fixed accuracy remains bounded as the filter order n is increased (see the numerical examples in [18]). Recall that for each iteration, the work is of order $O(n \log n)$ operations. Therefore, the work of obtaining the least squares estimator $\mathbf{w}(t)$ at each step t to a given accuracy is also of order $O(n \log n)$.

3. FFT-based sliding window RLS algorithm. As the FFT-based preconditioned conjugate gradient method is efficient in solving the normal equations (1.2), we employ the method in the sliding window RLS computations. In the following, we describe our FFT-based sliding window RLS algorithm. We assume that the sliding window RLS process is initialized, i.e., at time $t = M$, the Toeplitz matrices $T(M)$, $L(M)$, and $U(M)$ are given and the least squares estimator $\mathbf{w}(M)$ is also computed. Otherwise, the first step can be done by applying the conjugate gradient method to solving the preconditioned system

$$C(M)^{-1}[X(M)^*X(M)]\mathbf{w}(M) = C(M)^{-1}X^*(M)\mathbf{d}(M),$$

in factored form [7, 18], with the zero-vector as our initial approximation to the filter coefficient vector. The general step of our algorithm is described as follows.

ALGORITHM: FFT-BASED SLIDING WINDOW RLS ALGORITHM. Let M denote the length of the sliding data window, and let k denote the block length of the update and downdate blocks. The n-by-n Toeplitz matrix $T(t)$, the k-by-n downdating matrix $Z(t)$, the k-by-n updating matrix $Y(t)$, and the associated right-hand side M-vector $X(t)^*\mathbf{d}(t)$ at step t in the RLS process are given. Let $\mathbf{w}(t)$ denote the least squares estimator at step t. This algorithm computes the least squares estimator $\mathbf{w}(t+k)$ at step $t+k$.

- Generate information for the three n-by-n Toeplitz matrices $T(t+k)$, $L(t+k)$, and $U(t+k)$ and the associated right-hand side n-vector $X(t+k)^*\mathbf{d}(t+k)$, using the method proposed in §2.1.
- Find the first column of the optimal circulant preconditioner $C(t+k)$ for $T(t+k)$ using the method described in §2.2.
- Apply the conjugate gradient algorithm, as in §2.2, to solving the preconditioned system represented as

$$C(t+k)^{-1}[X(t+k)^*X(t+k)]\mathbf{w}(t+k) = C(t+k)^{-1}X(t+k)^*\mathbf{d}(t+k),$$

with starting initial guess $\mathbf{w}(t)$.

Next we provide some remarks about the algorithm:

1. The FFT-based sliding window RLS algorithm consists of three basic parts. The FFT can be used to generate normal equations and circulant preconditioners in the first and second parts. The number of operations required is $O(\max\{k, n\}\log n)$. In the third part, by using FFT, the cost of matrix-vector multiplications involving the matrices $C(t+k)^{-1}$, $T(t+k)$, $L(t+k)$, $L(t+k)^*$, $U(t+k)$, and $U(t+k)^*$ can be done efficiently in $O(n\log n)$ operations. It follows that the number of operations per iteration in the preconditioned conjugate gradient method is $O(n\log n)$. The total work of obtaining the least squares estimators at each step t to a given accuracy is of order $O(n\log n)$ operations for point-processing and $O(\max\{k, n\}\log n + n\log n)$ for block-processing. In the case of block-processing when $k \geq n$, the method on average will require only $O(\log n)$ operations per adaptive filter input. A corresponding scheme for averaging the computations over n steps has been considered for the LMS algorithm by Marshall and Jenkins [16].
2. The basic tool of our fast sliding window RLS algorithm is the FFT. Since the FFT is highly parallelizable and has been implemented on multiprocessors efficiently [1, p. 238], our algorithm can be expected to perform efficiently in a parallel environment for large-scale or for near real-time applications. One could, for example, assign each step of the algorithm to a different group of processors. A group of processors would be responsible for generation of the right-hand side vector $X(t)^*d(t)$ of the normal equations, and the first columns of the Toeplitz matrices $T(t)$, $L(t)$, and $U(t)$ at step t, respectively. Then a group of processors can be used to generate the first column of circulant preconditioner $C(t)$. The conjugate gradient method can be implemented on an another group of processors.
3. When the input stochastic process is stationary and the adaptive filter system is subjected to the effect of white noise, both the expected value of the least squares estimators $\mathbf{w}(t+k)$ and $\mathbf{w}(t)$ are equal to the solution of the discrete Wiener–Hopf equation, see [13, pp. 165–167]. Thus if k is not large, $\mathbf{w}(t)$ will be a good initial approximation (instead of a randomly chosen one) to the least squares estimator $\mathbf{w}(t+k)$ when the input process of the adaptive system is time varying in practical

applications. It follows that the number of iterations of the preconditioned conjugate gradient method can possibly be reduced with this choice of starting vector; see the numerical results in §4.

Next we demonstrate the convergence of the sliding window RLS algorithm involving multiple updating and downdating computations. We consider two aspects of the problem in the analysis: (1) the estimate $\mathbf{w}(t)$, and (2) the average least squares error of the RLS process. We use a probabilistic approach to consider these aspects of the problem. For the analysis, we first assume that the desired response $d(t)$ and the input sample column vector $\mathbf{x}(t)$ are related by a *multiple linear regression model.* In particular, we write

$$d(t) = e_0(t) + \mathbf{x}(t)^* \hat{\mathbf{w}}, \tag{3.1}$$

where $\hat{\mathbf{w}}$ is an n-by-1 constant regression parameter vector of the model and $e_0(t)$ is the measurement error. We also assume that the measurement error process $\{e_0(t)\}$ is white noise with zero mean and variance τ^2. In addition, we assume that the input process satisfies assumption (A). This assumption is equivalent to saying that the adaptive system operates in a stationary environment.

We first give the following lemma which is useful later in the analysis of the convergence of the sliding window RLS algorithm.

LEMMA 3.1. *Let $\{e_0(t)\}$ be a discrete-time white noise process where the mean of the process is equal to* 0 *and the variance of the process is equal to τ^2. If* $\mathrm{Var}(e_0(t)^2) = \nu \quad \forall t = 1, 2, \ldots,$ *then for any given $\epsilon > 0$,*

$$\Pr\left\{ \left\| \frac{1}{\sqrt{M}} \mathbf{e}_0(t) \right\|_2 \le \sqrt{\tau^2 + \epsilon^2} \right\} > 1 - \frac{\nu}{M\epsilon^4},$$

where $\mathbf{e}_0(t)$ is an M-vector given by

$$\mathbf{e}_0(t) = [e_0(t - M + 1), e_0(t - M + 2), \ldots, e_0(t)]^T \tag{3.2}$$

and $t = M, M + 1, \ldots.$

Proof. Since $\{e_0(t)\}$ is a white noise process, i.e., it is a sequence of uncorrelated random variables, the mean and the variance of the random variable $\frac{1}{M} \sum_{j=1}^{M} e_0(t - M + j)^2$ is equal to τ^2 and ν/M, respectively. By Chebyshev's inequality (see Fuller [10]), we get the following result:

$$\Pr\left\{ \left| \frac{1}{M} \sum_{j=1}^{M} e_0(t - M + j)^2 - \tau^2 \right| \le \epsilon^2 \right\} > 1 - \frac{\nu}{M\epsilon^4}.$$

The remaining results can be derived in a straightforward way by considering simple probability arguments. □

We let $\mathbf{a}(t)$ denote the weight-error vector at step t, defined as the difference between the least squares estimator $\mathbf{w}(t)$ produced by the sliding window RLS algorithm and the regression parameter vector $\hat{\mathbf{w}}$ of the multiple linear regression model, i.e.,

$$\mathbf{a}(t) \equiv \mathbf{w}(t) - \hat{\mathbf{w}}. \tag{3.3}$$

We first prove that with probability 1, $||\mathbf{a}(t)||_2$ is bounded by a constant depending on the variance of the process of the measurement error and minimum values of the spectral density function $f(\theta)$ of the input process.

THEOREM 3.2. *Let the input process satisfy assumption* (A). *Let the desired response* $d(t)$ *be given by* (3.1). *If the variance of* $e_0(t)^2$ *is bounded* $\forall t = 1, 2, \ldots$, *then for any given* $\epsilon > 0$ *and* $0 < \delta < 1$,

$$\Pr\left\{ \|\mathbf{a}(t)\|_2 \le \sqrt{\frac{\epsilon^2 + \tau^2}{f_{\min} - \epsilon}} \right\} > 1 - \epsilon, \tag{3.4}$$

provided that the number of data samples J *is sufficiently large* ($J \gg n$), *where* τ^2 *is the variance of the measurement error* $\{e_0(t)\}$.

Proof. We assume in §2.1 that the normal equations matrix $X(t)^*X(t)$ is non-singular. Thus the least squares estimator $\mathbf{w}(t)$ at step t is given by

$$\mathbf{w}(t) = [X(t)^*X(t)]^{-1}X(t)^*\mathbf{d}(t),$$

where $\mathbf{d}(t)$ is an M-vector given by (2.3). Therefore, by (3.1) and (3.3), we have

$$\mathbf{a}(t) = [X(t)^*X(t)]^{-1}X(t)^*\mathbf{e}_0(t),$$

where the vector $\mathbf{e}_0(t)$ is given by (3.2). It follows that

$$\begin{aligned} ||\mathbf{a}(t)||_2 &= \left(\frac{1}{J}X(t)^*X(t)\right)^{-1}\frac{1}{J}X(t)^*\mathbf{e}_0(t) \\ &\le \left\| \left(\frac{1}{J}X(t)^*X(t)\right)^{-1}\frac{1}{\sqrt{J}}X(t)^*\right\|_2 \left\|\frac{1}{\sqrt{J}}\mathbf{e}_0(t)\right\|_2 \\ &\le \sqrt{\lambda_{\min}\left(\frac{1}{J}X(t)^*X(t)\right)}\left\|\frac{1}{\sqrt{J}}\mathbf{e}_0(t)\right\|_2 . \end{aligned} \tag{3.5}$$

Then we note by Lemma 2.3 that

$$\Pr\left\{\lambda_{\min}\left(\frac{1}{J}X(t)^*X(t)\right) \ge f_{\min} - \epsilon\right\} > 1 - \delta,$$

where $f_{\min}$ are the maximum and the minimum values of the positive spectral density function $f(\theta)$ of the input stationary process. Combining the above result with Lemma 3.1, we have

$$\Pr\left\{ ||\mathbf{a}(t)||_2^2 \le \sqrt{\frac{\epsilon^2 + \tau^2}{f_{\min} - \epsilon}} \right\} > 1 - \frac{\nu}{J\epsilon^4} - 2\delta, \tag{3.6}$$

where ν is the variance of $e_0(t)$. The theorem follows by using simple probability arguments in (3.6), provided that J is sufficiently large. □

We see from the bound (3.4) that the ℓ_2 norm of the weight-error vector is magnified by the smallest eigenvalue of $X(t)^*X(t)$. Thus the sensitivity of the sliding window RLS algorithm is determined initially by the conditioning of the normal equations matrix $X(t)^*X(t)$.

Next, we consider the average least squares error of the sliding window RLS algorithm. For the sliding window RLS method, we solve the least squares problem (2.4), and thus the average least squares error is given by (2.2). In the following analysis, we let $\zeta(t)$ denote the average least squares error of the sliding window RLS estimate at each step t, i.e.,

$$\zeta(t) = \frac{1}{J}\sum_{j=t-M+1}^{t} |e(j)|^2.$$

Recall that J is the number of data samples taken in the sliding window RLS method at step t. We will show that with probability 1, the average least squares error $\zeta(t)$ is bounded by a constant depending on the variance of the process of the measurement error and the maximum and minimum values of spectral density function $f(\theta)$ of the input process, provided that J is sufficiently large.

THEOREM 3.3. *Let the input process satisfy assumption (A). If the desired response $d(t)$ is given by* (3.1), *then for any given $\epsilon > 0$ and $0 < \delta < 1$,*

$$\Pr\left\{ |\zeta(t)| \le (\tau^2+\epsilon^2)\left(1+\sqrt{\frac{f_{\max}+\epsilon}{f_{\min}-\epsilon}}\right)^2 \right\} > 1-\delta,$$

provided that the number of data samples J is sufficiently large $(J \gg n)$.

Proof. By (3.1) and (3.3), we can write

$$\begin{aligned}
|\zeta(t)| &= \frac{1}{J}\,\|\mathbf{d}(t) - X(t)\mathbf{w}(t)\|_2^2 \\
&= \frac{1}{J}[\mathbf{d}(t) - X(t)\mathbf{w}(t)]^*[\mathbf{d}(t) - X(t)\mathbf{w}(t)] \\
&= \frac{1}{J}\mathbf{e}_0^*(t)\mathbf{e}_0(t) + \frac{1}{J}\mathbf{a}^*(t)X(t)^*X(t)\mathbf{a}(t) + \frac{1}{J}\mathbf{e}_0^*(t)X(t)\mathbf{a}(t) + \frac{1}{J}\mathbf{a}^*(t)X(t)^*\mathbf{e}_0(t) \\
&\le \left\|\frac{1}{\sqrt{J}}\mathbf{e}_0(t)\right\|_2^2 + \left\|\frac{1}{J}X(t)^*X(t)\right\|_2 \|\mathbf{a}(t)\|_2^2 \\
&\quad +2\left\|\frac{1}{\sqrt{J}}\mathbf{e}_0(t)\right\|_2 \left\|\frac{1}{J}X(t)^*X(t)\right\|_2^{\frac{1}{2}} \|\mathbf{a}(t)\|_2 . \qquad (3.7)
\end{aligned}$$

Combining Theorem 3.2, Lemma 2.3, and Lemma 3.1, the theorem follows by considering probability arguments and using (3.7). □

Based on Theorems 3.2 and 3.3, it follows that the sliding window RLS algorithm always converges when the adaptive system is operated in a stationary environment.

4. Numerical experiments. In this section, numerical experiments are reported to illustrate the convergence performance of the FFT-based sliding window RLS algorithm. All the computations are done using Matlab on an HP-715 workstation. We first illustrate the convergence rate of the preconditioned conjugate gradient method by using adaptive finite impulse response (FIR) system identification computations. FIR system identification has wide applications in engineering [13]. Figure 4.1 is a block diagram for the implementation of our algorithm in an adaptive FIR system identification model. The input signal $x(t)$ drives the unknown system to produce the output sequence $d(t)$. We model the unknown system as an FIR filter. If the unknown system is actually an FIR system, then the model is exact.

We apply our method to first and second order autoregressive models as input stochastic processes. The first order autoregressive AR(1) and second order autoregressive AR(2) processes are given by

$$x(t) + \rho x(t-1) = v(t) \quad \text{and} \quad x(t) + \chi_1 x_{t-1} + \chi_2 x_{t-2} = v(t)$$

[22, pp. 238, 241], respectively, where $\{v(t)\}$ is a white noise process with variance η^2. In our tests, we choose the parameters in each input process so that the corresponding spectral density functions $f(\theta)$ are positive and in the Wiener class. The variance η^2 of $\{v(t)\}$ is chosen as 1. The reference (unknown) system is an nth order linear phase FIR filter with

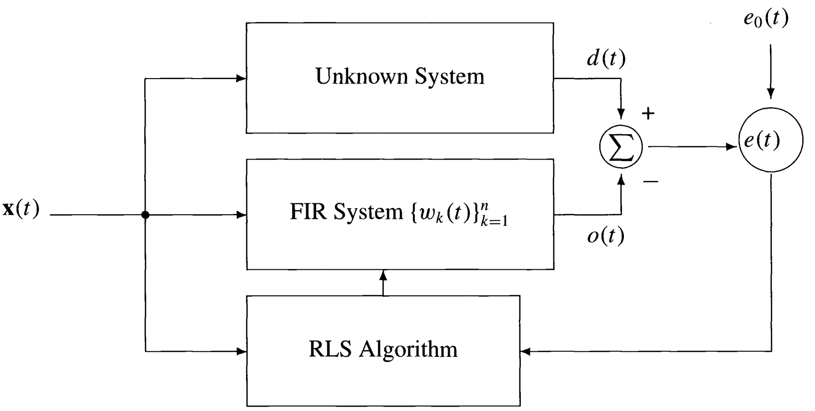


FIG. 4.1. *Adaptive FIR system identification model.*

uncorrelated Gaussian white noise added. The finite impulse response $\{\hat{w}_k\}_{k=1}^n$ used for the unknown system is given by

$$\hat{w}_k = 1.1 - \frac{|2k-n-1|}{n-1}, \quad k = 1, 2, \ldots, n.$$

We note that the shape of the FIR filter is triangular. The Gaussian white noise measurement error $\{e_0(t)\}$ with variance τ^2 is added into the FIR system identification model to test the performance of the RLS algorithm. In the figures below, $N = M/n$ is the number of blocks of the data samples used in data matrix $X(t)$, where the size of each block is equal to n. Recall that M is the length of the sliding window RLS. In the numerical tests, the stopping criterion of the preconditioned conjugate gradient method is that the ℓ_2 norm of the residual vector after j iterations is less than 10^{-7}.

We first consider the number of iterations at each recursive updating and downdating step of the FFT-based sliding window RLS algorithm. We use $k = 1$ as rank numbers for the updating and downdating computations to test our method. Figures 4.2 and 4.3 show the average numbers of iterations of the normal systems and of the preconditioned systems at each adaptive time step, averaged over 20 runs of the algorithm. In the tests, we use zero vector and current filter coefficient vectors as our initial guesses at time step $t = M$ and at time step $t = M + k, M + 2k, \ldots$, respectively. From the numerical results, we see that the conjugate gradient algorithm for our preconditioned systems converges very quickly, and the number of iterations required for convergence is always less than that of nonpreconditioned systems. Figure 4.4 shows how the choice of the initial vectors in the preconditioned conjugate gradient method improves the convergence. In these tests, the current solution vectors and randomly chosen vectors are used as initial guesses at each adaptive time step to test the performance of the algorithm. From the numerical results, we see that when $k = 1$, the reduction in the number of iterations required for convergence in preconditioned systems at each updating and downdating step is significant. However, when $k = n$, the reduction is not as significant as $k = 1$. This trend is not surprising, since $\mathbf{w}(t)$ is expected to be a less effective starting vector at time $t + k$, for larger k.

Next, we consider the ℓ_2 norm of the weight-error vector $\mathbf{a}(t)$ and the average least square error $\zeta(t)$ at each recursive step to test the convergence behavior of our algorithm. Figure 4.5 shows the ℓ_2 norm of the weight-error vector and the average least square error for AR(1) with $\rho = -0.9999$ as input stationary processes at each recursive updating and downdating step. In these cases, the variances of the Gaussian white noise measurement error $\{e_0(t)\}$ are equal

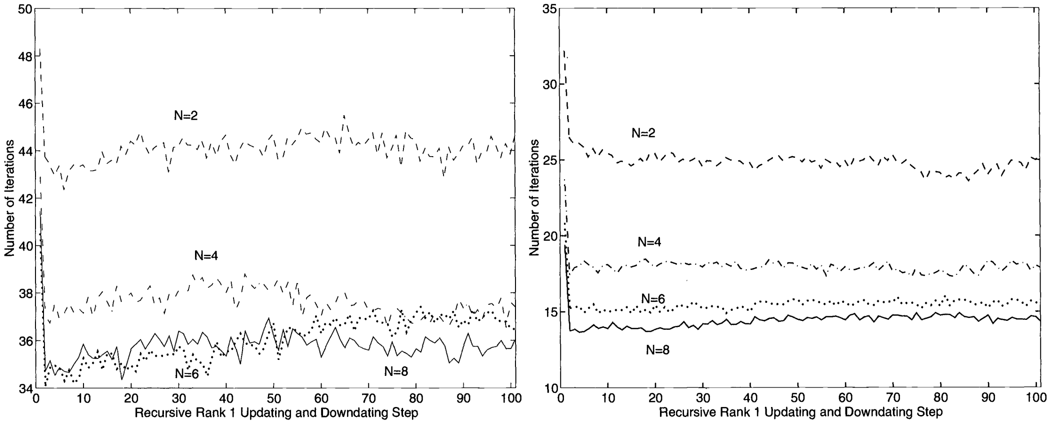


FIG. 4.2. (*Left*) AR(1) *input process when* $n = 32$, $\rho = -0.9999$, *variance of noise* = 0.025, *and no preconditioner is used.* (*Right*) AR(1) *input process when* $n = 32$, $\rho = -0.9999$, *variance of noise* = 0.025, *and FFT-based preconditioner is used.*

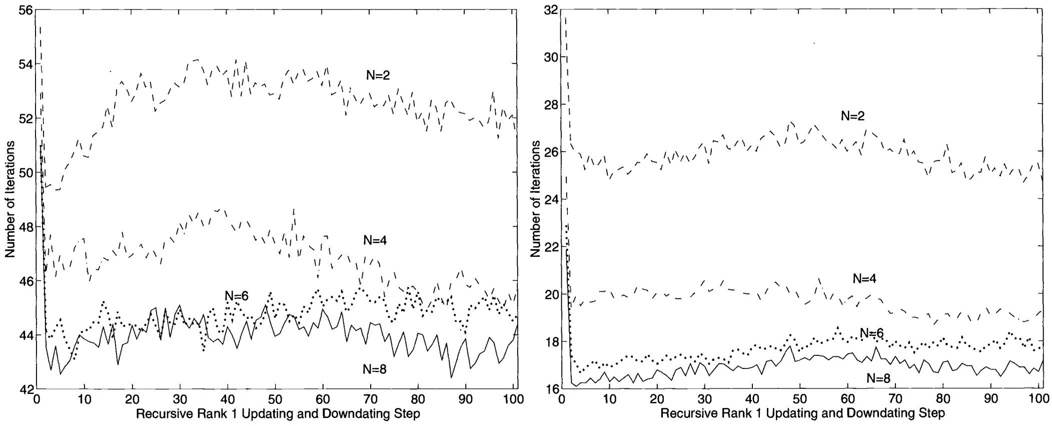


FIG. 4.3. (*Left*) AR(2) *input process when* $n = 32$, $\chi_1 = -1.4$, $\chi_2 = 0.45$, *variance of noise* = 0.025, *and FFT-based preconditioner is used.* (*Right*) AR(2) *input process when* $n = 32$, $\chi_1 = -1.4$, $\chi_2 = 0.45$, *variance of noise* = 0.025, *and FFT-based preconditioner is used.*

to 0.025. We remark that the numerical results are average values based on 20 runs of our algorithm. From the figures, we note that both $||\mathbf{a}(t)||_2$ and $\zeta(t)$ are bounded by a constant depending on the variances of the background white noise. We also see that as $M = N * n$ increases, the average least squares error $|\zeta(t)|$ increases; however, $||\mathbf{a}(t)||_2$ decreases.

The latter phenomenon can be explained by considering the expected value of the weight-error correlation matrix $K(t)$ defined by

$$K(t) \equiv \mathcal{E}(\mathbf{a}(t)\mathbf{a}(t)^*)$$

at step t. Haykin [13, pp. 487–489] proved that the least squares estimator $\mathbf{w}(t)$ is unbiased, i.e.,

$$\mathcal{E}(\mathbf{w}(t)) = \hat{\mathbf{w}},$$

and the expected value of $K(t)$ is given by

$$K(t) = \tau^2[X(t)^*X(t)]^{-1}, \tag{4.1}$$

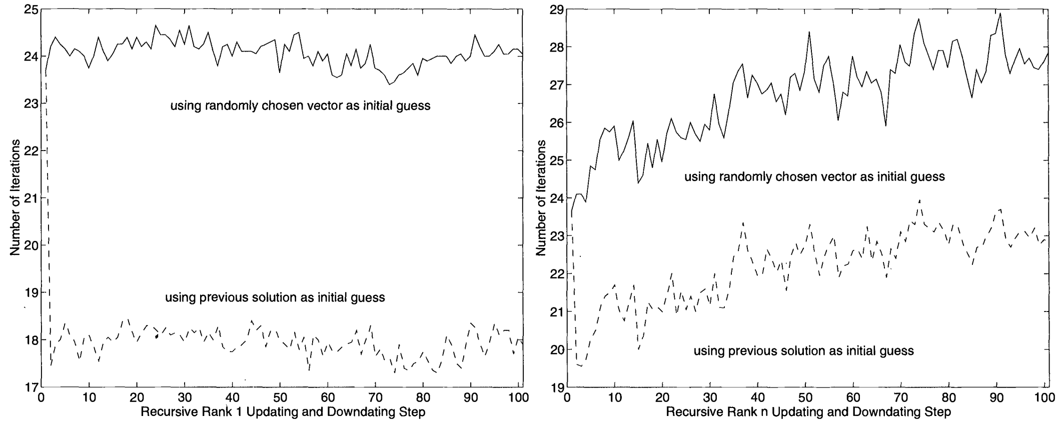


FIG. 4.4. *(Left)* AR(1) *input process when* $n = 32$, $\rho = -0.9999$, *variance of noise* $= 0.025$, *and FFT-based preconditioner is used. (Right)* AR(1) *input process when* $n = 32$, $\rho = -0.9999$, *variance of noise* $= 0.025$, *and FFT-based preconditioner is used.*

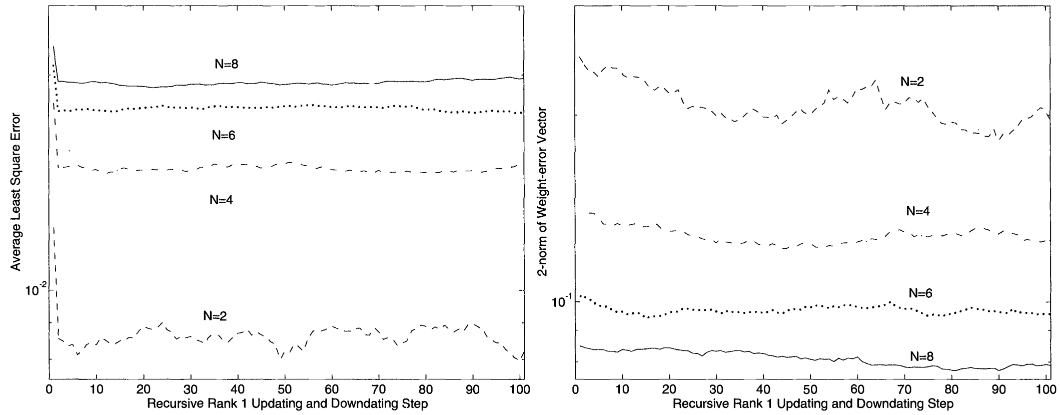


FIG. 4.5. *(Left)* AR(1) *input process when* $n = 32$, $\rho = -0.9999$, *and variance of noise* $= 0.025$. *(Right)* AR(1) *input process when* $n = 32$, $\rho = -0.9999$, *and variance of noise* $= 0.025$.

where τ^2 is the variance of the measurement error $\{e_0(t)\}$. We can rewrite (4.1) as follows:

$$K(t) = \frac{\tau^2}{J}\left[\frac{X(t)^*X(t)}{J}\right]^{-1}.$$

From Lemma 2.3 we observe that with probability 1, $||K(t)||_2$ is bounded by

$$||K(t)||_2 \le \frac{\tau^2}{J(f_{\min} - \epsilon)},$$

provided that J is sufficiently large. Based on the result about the correlation matrix of the weight-error vector, we will expect $||\mathbf{a}(t)||_2$ to decrease as J increases. Moreover, Figure 4.6 shows the effect of background white noise on $||\mathbf{a}(t)||_2$ and $|\zeta(t)|$. Different levels of variances τ^2 of the Gaussian white noise are used. We see that both $||\mathbf{a}(t)||_2$ and $|\zeta(t)|$ increase as τ^2 increases.

In the following tests we compare our FFT-based sliding window RLS algorithm with standard recursive least squares $O(n^2)$ operations [13, p. 485] and fast transversal filter $O(n)$

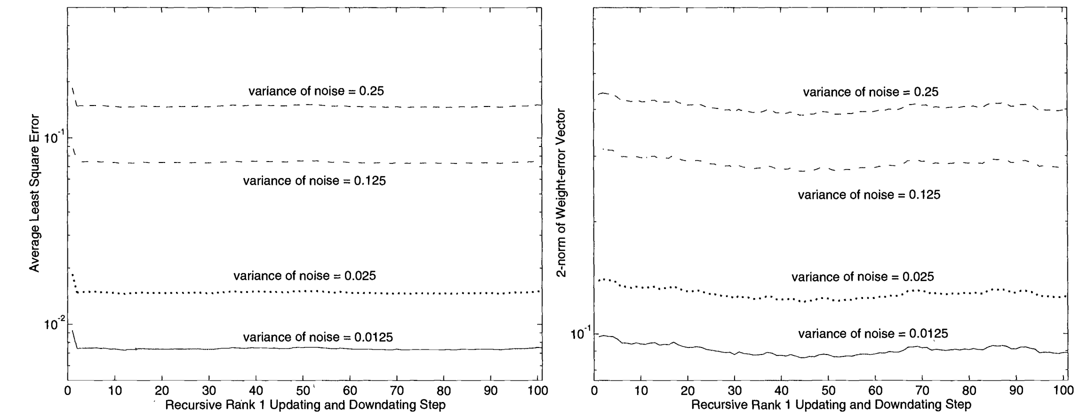


FIG. 4.6. (*Left*) AR(1) *input process when* $n = 32$, $\rho = -0.9999$, *and* $N = 4$. (*Right*) AR(1) *input process when* $n = 32$, $\rho = -0.9999$, *and* $N = 4$.

operations algorithms [13, pp. 586–599]. An exponential weighting factor γ is generally used in the standard recursive least squares and fast transversal filter algorithms. The inverse of $1-\gamma$ is approximately related to a "measure" of the memory of the algorithm. Here, the length of the sliding window M used in our FFT-based sliding window RLS algorithm is related to γ by the approximation formula,

$$M \approx \frac{1}{1-\gamma}.$$

Figures 4.7 and 4.8 show the prior average least squares error $(d(t) - \mathbf{w}^*(t-1)\mathbf{x}(t))^2$ and the ℓ_2 norm of weight-error vector of different adaptive filter algorithms when an AR(1) input process with $\rho = -0.9999$ is used. Different levels of variances of the Gaussian white noise are used to test the performances of different adaptive filter algorithms. We see from Figures 4.7(b) and 4.8(b) that the fast transversal filter algorithm given in [13, pp. 586–599] does not converge when noise is added to the adaptive FIR system. However, both the standard recursive least squares (Figures 4.7(a) and 4.8(a)) and our FFT-based sliding window (Figures 4.7(c) and 4.8(c)) algorithms converge very quickly.

Next, we compare the complexity of our algorithm with the standard recursive least squares algorithm. Figures 4.9 and 4.10 show the number of kilo-flops used by the standard recursive least squares algorithm and the FFT-based preconditioned conjugate gradient method for rank 1, rank 4, rank 10, and rank n recursive updating. We see that when the order of the filter is small, the complexity of our algorithm is greater than that of the standard recursive least squares algorithm. However, when the order of the filter is large, the complexity of our algorithm is quite competitive with the standard least squares algorithm. From Figures 4.9 and 4.10, we note that the reduction in the complexity of our algorithm is significant for rank n recursive updating. Thus our algorithm is particularly useful in signal processing applications where the sizes of the filters are very large. (For example, in the case of acoustic echo or active noise cancellation problems, n is between 250 and 2000, depending on the precise application [17], [20].) In addition, more effective parallel implementations of our FFT-based scheme are possible [21].

Finally, we remark that our FFT-based sliding window RLS scheme also facilitates tracking the smallest and largest eigenvalues of the normal equations matrix $X(t)^*X(t)$ at each time step. The procedure is to determine the optimal circulant approximation to the normal

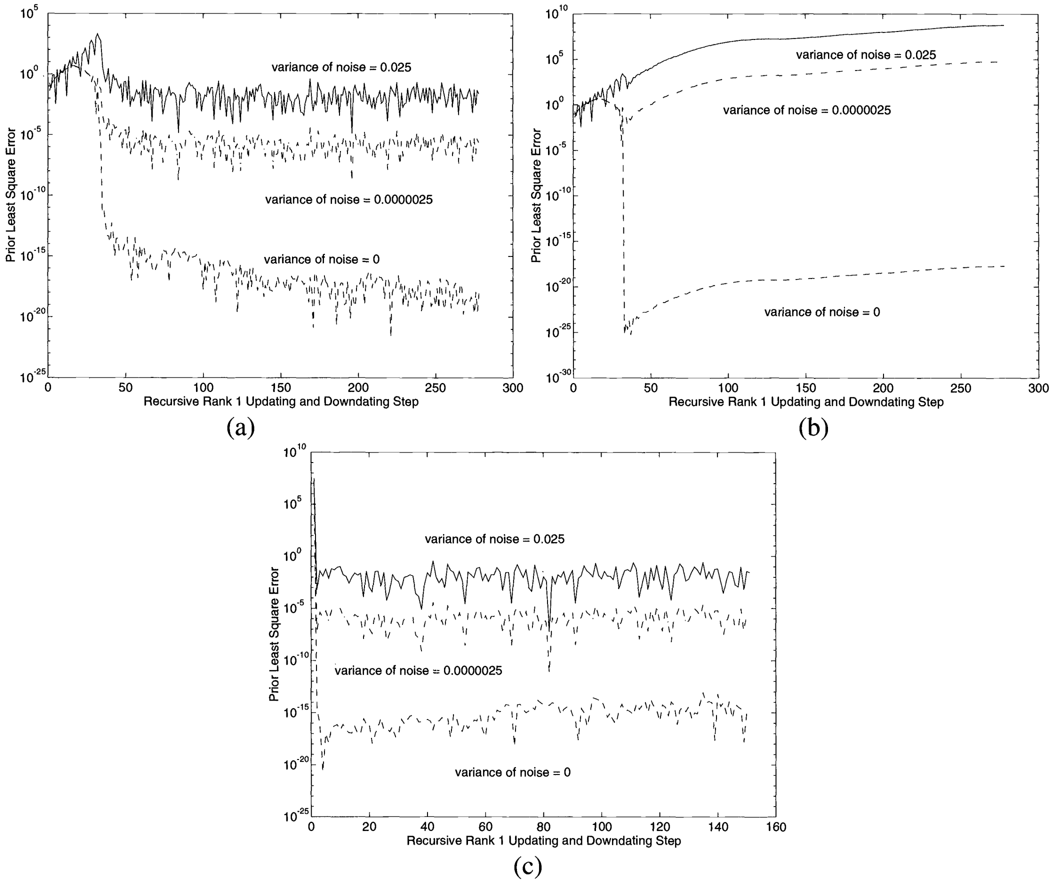


FIG. 4.7. (a) *Standard recursive least squares algorithm with* $\gamma = 0.9922$. (b) *Fast transversal filter algorithm with* $\gamma = 0.9922$. (c) *FFT-based sliding window RLS algorithm with* $N = 4$.

equations matrix. We note that by (2.4) and linearity of the optimal circulant approximation, we have

$$||C - X(t)^*X(t)||_F \leq ||C(t) - T(t)||_F + ||C_L(t) - L(t)^*L(t)||_F + ||C_U(t) - U(t)^*U(t)||_F,$$

where we define

$$C = C(t) - C_L(t) - C_U(t). \tag{4.2}$$

Here circulant matrices $C(t)$, $C_L(t)$, and $C_U(t)$ are the optimal circulant preconditioners of the matrices $T(t)$, $L(t)^*L(t)$, and $U(t)^*U(t)$ respectively. By Lemma 2.1, we have

$$\lambda_{\min}(X(t)^*X(t)) \leq \lambda_{\min}(C) \leq \lambda_{\max}(C) \leq \lambda_{\max}(X(t)^*X(t)). \tag{4.3}$$

By considering the spectral condition number $\kappa(C(t))$, it follows from (4.3) that

$$\kappa(C(t)) = \frac{\lambda_{\max}(C(t))}{\lambda_{\min}(C(t))} \leq \kappa(X(t)^*X(t)), \quad \text{or} \quad \sqrt{\kappa(C(t))} \leq \kappa(X(t)). \tag{4.4}$$

Here $\kappa(\cdot)$ denotes the spectral condition number of a matrix. Thus (4.4) provides a lower bound on the spectral condition number of $X(t)^*X(t)$. For the generation of the circulant matrices

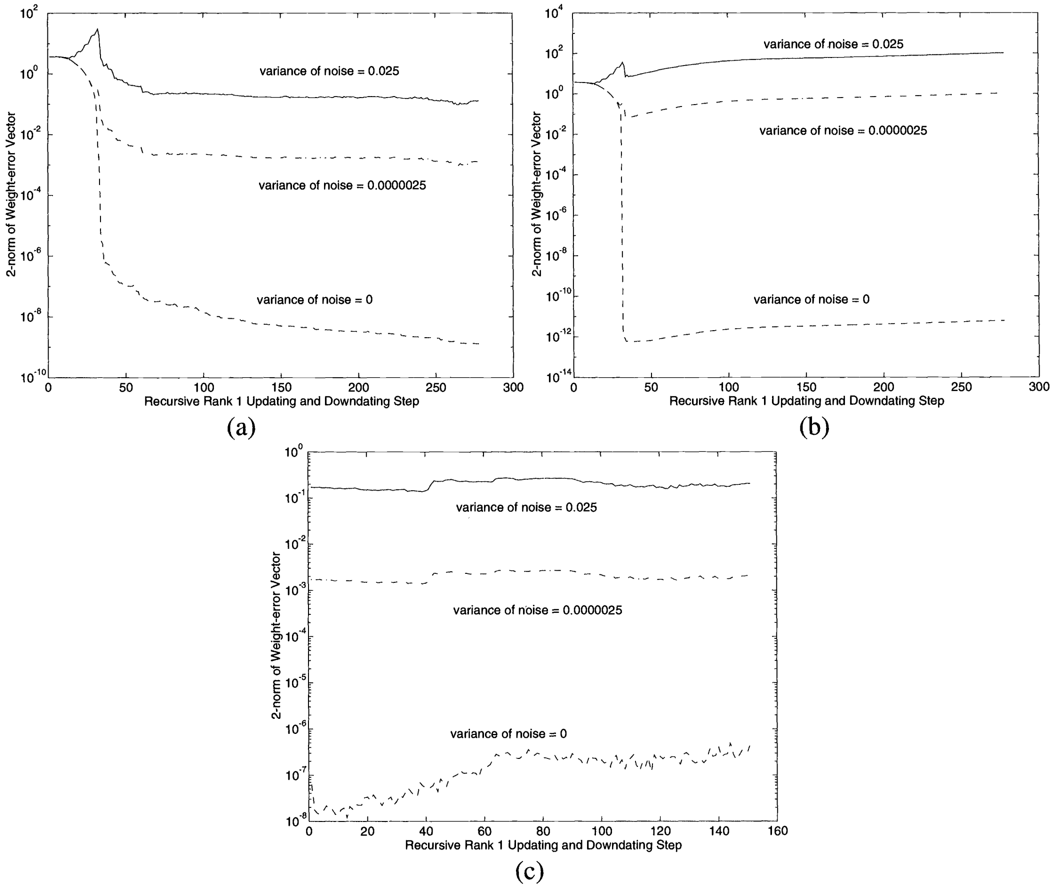


FIG. 4.8. (a) *Standard recursive least squares algorithm with* $\gamma = 0.9922$. (b) *Fast transversal filter algorithm with* $\gamma = 0.9922$. (c) *FFT-based sliding window RLS algorithm with* $N = 4$.

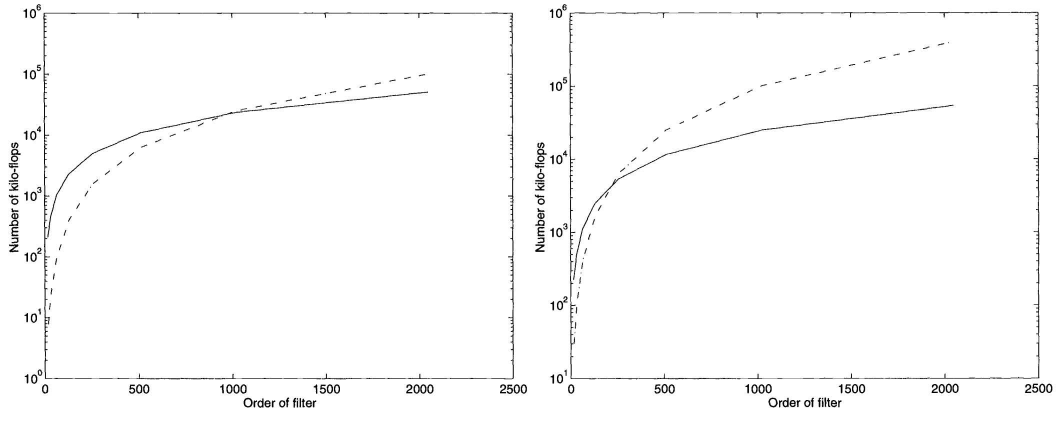


FIG. 4.9. (*Left*) *Rank* 1 *updating when* $N = 4$ *and* AR(1) *input process with* $\rho = -0.3$ *is used.* (*Right*) *Rank* 4 *updating when* $N = 4$ *and* AR(1) *input process with* $\rho = -0.3$ *is used. Dashed curve: Standard RLS. Solid curve: our FFT-based PCG.*

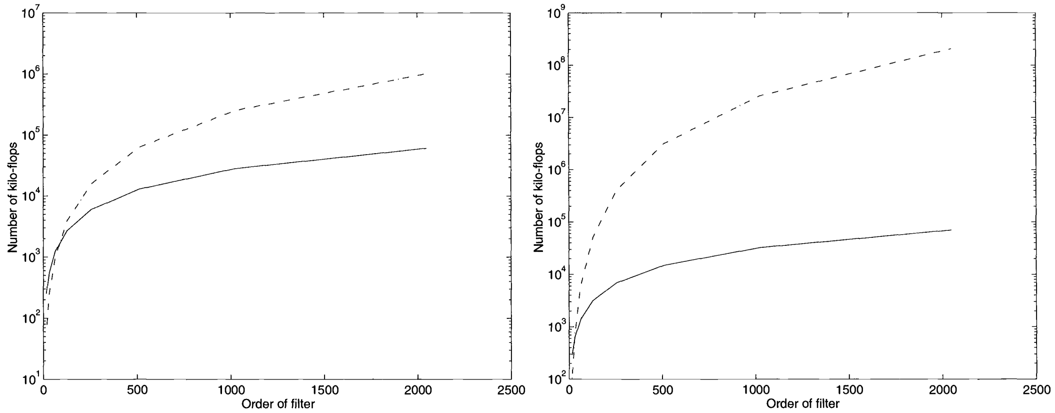


FIG. 4.10. (*Left*) *Rank* 10 *updating when* $N = 4$ *and* AR(1) *input process with* $\rho = -0.3$ *is used.* (*Right*) *Rank* n *updating when* $N = 4$ *and* AR(1) *input process with* $\rho = -0.3$ *is used. Dashed curve: Standard RLS. Solid curve: our FFT-based PCG.*

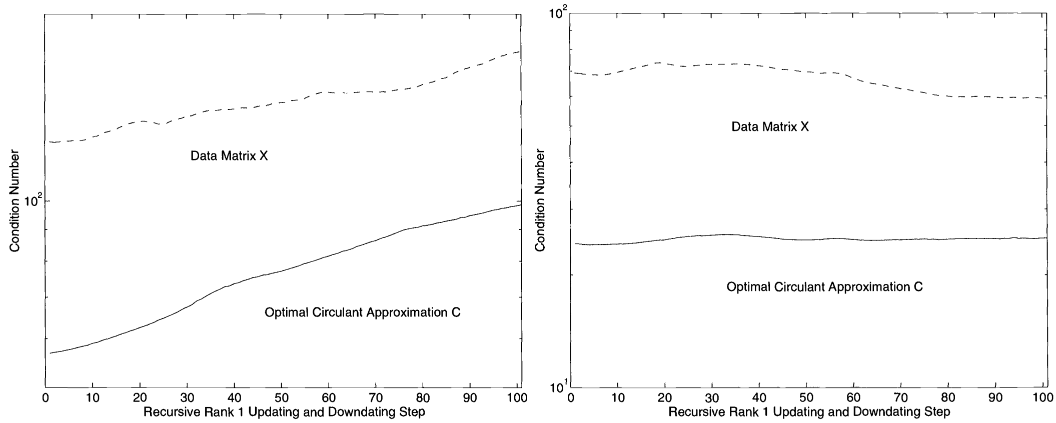


FIG. 4.11. (*Left*) AR(1) *input process when* $n = 32$, $\rho = -0.9999$, *and* $N = 4$. (*Right*) AR(2) *input process when* $n = 32$, $\chi_1 = -1.4$, $\chi_2 = 0.45$, *and* $N = 4$.

$C_L(t)$ and $C_U(t)$, Chan, Jin, and Yeung [5] have proved that the circulant matrices $C_L(t)$ and $C_U(t)$ can be generated in $O(n \log n)$ operations. It follows that the eigenvalues of C can be obtained in $O(n \log n)$ operations. Consequently, if (4.4) is large, then the data matrix $X(t)$ is ill-conditioned. Regularization schemes can then be used to stabilize the computations in the FFT-based sliding window RLS algorithm if $X(t)$ is ill-conditioned. In particular, Hanke, Nagy, and Plemmons [12] have devised a regularization scheme based on modifying the eigenvalues of $C(t)$ that can be applied to overcome the effects of ill-conditioning in $X(t)$. In addition, other regularization techniques for Toeplitz least squares problems have also been studied by Chan, Nagy, and Plemmons [7, 8]. These approaches using regularization can lead to stable and robust implementations of our FFT-based sliding window RLS algorithm. Figure 4.11 lists the corresponding average spectral condition number of the circulant matrix C and the data matrix $X(t)$ (symbol X) at each recursive updating and downdating step. We note that the condition number for C provides a reasonable lower bound as it adaptively tracks the changes in the condition number of the data matrix at each time step.

5. Concluding remarks. In this paper we have proposed a new FFT-based sliding window recursive least squares algorithm. Preliminary results show that, for many important problems, FFT-based iterative methods can compete with direct methods in an adaptive signal processing environment. A summary of some of these results is also included in [19].

Also, we remark that our iterative FFT-based sliding window algorithm can be adapted to handle two-dimensional (2-D) signal processing applications. Here direct methods are well known to experience numerous difficulties (see, e.g., [13]). Chan, Nagy, and Plemmons [8] have studied block least squares computations, min $\|b - Tx\|_2$, where T is a rectangular Toeplitz-block matrix. They consider there the solution of block least squares problems by the preconditioned conjugate gradient algorithm using square nonsingular circulant-block and related preconditioners, constructed from the blocks of the rectangular matrix T. Preconditioning with such matrices allows efficient implementation using the 2-D FFT. This extends the earlier work on preconditioners for Toeplitz least squares iterations for 1-D problems outlined in §2. Our iterative FFT-based sliding window RLS algorithm described in §3 can thus be applied to 2-D signal processing applications, where the data matrix $X(t)$ generally has a Toeplitz-block structure.

Acknowledgment. We would like to thank the referees for their valuable comments.

REFERENCES

[1] S. Aki, *The Design and Analysis of Parallel Algorithms*, Prentice-Hall International Inc., London, 1989.

[2] S. Alexander, *Adaptive Signal Processing, Theory and Applications*, Springer-Verlag, New York, 1986.

[3] P. J. Brockwell and R. A. Davis, *Time Series: Theory and Methods*, Springer-Verlag, New York, 1987.

[4] R. Chan and G. Strang, *Toeplitz equations by conjugate gradients with circulant preconditioner*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 104–119.

[5] R. Chan, X. Jin, and M. Yeung, *The circulant operator in the Banach algebra of matrices*, Linear Algebra Appl., 149 (1991), pp. 41–53.

[6] R. Chan and M. Yeung, *Circulant preconditioners constructed from kernels*, SIAM J. Numer. Anal., 29 (1992), pp. 1093–1103.

[7] R. Chan, J. Nagy, and R. Plemmons, *Circulant preconditioned Toeplitz least squares iterations*, SIAM J. Matrix Anal. and Applic., 15 (1994), pp. 80–97.

[8] R. Chan, J. Nagy, and R. J. Plemmons, *FFT-based preconditioners for Toeplitz-block least squares problems*, SIAM J. Numer. Anal., 30 (1993), pp. 1740–1768.

[9] T. Chan, *An optimal circulant preconditioner for Toeplitz systems*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 766–771.

[10] W. Fuller, *Introduction to Statistical Time Series*, John Wiley, New York, 1976.

[11] G. Golub and C. Van Loan, *Matrix Computations*, 2nd Ed., The Johns Hopkins University Press, Baltimore, MD, 1989.

[12] M. Hanke, J. Nagy, and R. Plemmons, *Preconditioned iterative regularization for ill-posed problems*, Numerical Linear Algebra and Scientific Computing, L. Reichel, A. Ruttan, and R. Varga, eds., de Gruyter Press, Berlin, 1993, pp. 141–163.

[13] S. Haykin, *Adaptive Filter Theory*, 2nd Ed., Prentice-Hall, Englewood Cliffs, NJ, 1991.

[14] D. Kim and W. Alexander, *Stability analysis of the fast RLS algorithms*, Proc. IEEE Conf. on Acoustics, Speech & Signal Processing ICASSP-88, 1988, pp. 1361–1364.

[15] X. Luo and S. Qiao, *An error analysis of the fast RLS algorithms*, Rept. no. 231, Comm. Res. Lab., McMaster Univ., Hamilton, Ontario, Canada, 1991, under revision.

[16] D. Marshall and K. Jenkins, *A fast quasi-Newton adaptive filtering algorithm*, IEEE Trans. Signal Processing, 40 (1992), pp. 1652–1662.

[17] M. Montazeri and P. Duhamel, *A Set of Algorithms Linking NLMS and Block RLS Algorithms*, preprint, 1993.

[18] M. Ng and R. Chan, *Fast iterative methods for least squares estimations*, Numerical Algorithms, 6 (1994), pp. 353–378.

[19] M. Ng and R. Plemmons, *Fast Recursive Least Squares Using the FFT*, in Mathematics in Signal Processing III, J. McWhirter, ed., Claredon Press, Oxford, 1994, pp. 97–130.

[20] T. Petillon, A. Gilloire, and S. Theodoridis, *The Fast Newton Transversal Filter: An Efficient Scheme for Acoustic Echo Cancellation in Mobile Radio*, IEEE Trans. Signal Processing, 42 (1994), pp. 509–518.

[21] R. Plemmons, *FFT-based RLS in Signal Processing*, Proc. IEEE Confer. on Acoustics, Speech & Signal Processing ICASSP-93, 3 (1993), pp. 571–574.

[22] M. Priestley, *Spectral Analysis and Time Series*, Academic Press, New York, 1981.

[23] S. Qiao, *Error Propagation of Some Fast RLS Algorithms*, in Advanced Signal Processing Algorithms, Architectures, and Implementations IV, Franklin T. Luk, ed., 2027 (1993), pp. 448–454.

[24] L. Sibul, *Adaptive Signal Processing*, IEEE Press, New York, 1987.

[25] E. Tyrtyshnikov, *Optimal and Super-optimal Circulant Preconditioners*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 459–473.

[26] K. Zhao, F. Ling, H. Lev-Ari, and J. Proakis, *QRD-based Sliding Window Adaptive LS Lattice Algorithm*, Proc. IEEE Confer. on Acoustics, Speech & Signal Processing ICASSP-92, 3 (1992).

[27] ———, *Sliding Window Order–recursive Least Squares Algorithms*, preprint (1993).

SIAM J. SCI. COMPUT.
Vol. 17, No. 4, pp. 942–955, July 1996

A CONVOLUTION ALGORITHM WITH APPLICATION TO DATA ASSIMILATION*

RANJIT M. PASSI†, R. KENT GOODRICH‡, MARK LIMBER‡, AND JOHN C. DERBER§

Abstract. A computationally efficient algorithm to approximate a convolution $w * f$ is derived when the real-valued weighting functions w are dimensionally separable, i.e., $w(x_1, x_2) = w_1(x_1)w_2(x_2)$ and $w(x_k) = w(-x_k)$, and the function $f(i, j)$ is defined on a discrete integer lattice in $\mathcal{R}^2$. The algorithm consists of a product of operator polynomials $P_k(D_k)$, $k = 1, 2$, which are composed of simple averaging operators D_k operating on f. It generalizes the smoothing algorithm of Goodrich, Passi, and Limber [*Proc. 24th Symposium on the Interface: Computing Science and Statistics*, 1992] for data constrained to a uniformly spaced, large-dimensioned orthogonal grid lying in a two-dimensional space, using a Gaussian weighting function $w(r) = \exp(-r^2/b^2)$.

The new algorithm has a direct application to data assimilation problems in meteorology and oceanography. Data assimilation optimally combines output of a numerical model with observational data to derive more accurate initial conditions needed for numerical integration of the model equations. Computer simulations with the algorithm were found to be about ten times faster than a comparable algorithm of Derber and Rosati [*J. Phys. Ocean*, (1989), pp. 1333–1347].

Key words. operator polynomials, Gaussian function, separable functions, data assimilation, product polynomial algorithm

AMS subject classification. 65N60

1. Introduction.

Nowcasting and forecasting are important activities in meteorology. Nowcasting combines the output of a numerical forecast model with observations to provide dynamically balanced initial conditions used by the model to calculate the forecast. Now that the numerical models of the general circulation of the ocean have matured, these two activities are playing an important role in oceanography also to provide a realistic description of the various oceanic regions.

To describe adequately the general circulation of the ocean requires a fine spatial resolution and extremely small time steps for model integration. This combination leads to large in-core computer memory and computer processing time for the model integration. In addition, the numerical models have to be coupled with data assimilation schemes to derive accurate initial conditions for computing model forecasts. To make the nowcasting and forecasting practical, it is necessary that numerical modeling and data assimilation schemes be computationally efficient and not require inordinately large computer in-core memories.

The nowcasting step involves two separate operations of (i) optimally melding model output with observations (data assimilation), which is followed by (ii) dynamically balancing the melded output so that transients are not introduced in subsequent model integration. In this paper, we are mainly concerned with data assimilation where one combines the model output with the observations using a linear least squares method. Because of the large dimensions involved, the minimization in the least squares procedure is performed by the method of steepest descent, as it avoids inversion of the covariance matrix Σ_m of the model output (Derber and Rosati [5], hereafter, referred to as DR). The minimization procedure requires several matrix multiplications of the type $\Sigma_m \mathbf{g}$, where Σ_m stays fixed but vector $\mathbf{g}$ is varying. Because the dimensions of Σ_m are large and in-core memory parsimony is a necessity, Σ_m is

*Received by the editors September 14, 1993; accepted for publication (in revised form) March 15, 1995. This research was supported by the Office of Naval Research grant N00014-92-J-4112.

†Center for Ocean & Atmospheric Modeling, Stennis Space Center, MS 39529 (passi@coam.usm.edu).

‡Department of Mathematics, University of Colorado, Boulder, CO 80309 (goodrich@euclid.colorado.edu, marlim@auto-trol.com).

§National Weather Service/NOAA, World Weather Building, Washington, DC 20233 (wd23jd@sun1.wwb.noaa.gov).

not stored in-core. For a Gaussian covariance function, DR developed an algorithm to evaluate $\Sigma_m \mathbf{g}$ that was computationally efficient and avoided a direct evaluation and storage of Σ_m. This was achieved by repeated applications of a Laplacian operator.

In this paper, we develop a new algorithm (the product-polynomial operator (PPO) algorithm) that, while maintaining the accuracy, approximates $\Sigma_m \mathbf{g}$ more efficiently than the Laplacian algorithm; the approximation is affected by applying a product of operators that are polynomials in simple averaging operators. In addition, it is shown to admit, with equal ease and efficiency, a wider class of covariance functions which can be expressed as a product of factors that are even functions in a single dimension. The theory is developed in $\mathcal{R}^2$, but is easily generalized to higher dimensions. Finally, using computer simulations, our approximation is compared with the actual matrix multiplication, $\Sigma_m \mathbf{g}$, and with the DR algorithm.

In §2 we first develop the statistical framework needed to arrive at a common formulation of the optimum interpolation problem, leading to the least squares minimization solution by the steepest descent, iterative process, and the step where the matrix multiplication $\Sigma_m \mathbf{g}$ has to be evaluated. This is followed, in §3, by the development and description of the PPO algorithm for approximating $\Sigma_m \mathbf{g}$, along with a discussion on the determination of the operator-polynomial degrees, approximation accuracy, and implementation procedures. Although our emphasis is on the data assimilation aspect, in §4 we describe some steps that maintain dynamical balance in the different fields and minimize the effects of transients in our ocean data assimilation experiments. Finally, in §5, we present the results of numerical simulations that compare the performance of the PPO and the DR algorithms.

2. Statistical formulation of the data assimilation problem. The purpose of data assimilation is to estimate the true state of the ocean $\boldsymbol{\Theta}$ by combining the model output $\mathbf{T}_m$ and observations $\mathbf{T}_o$. The combined estimator (after dynamic initialization, in some cases; see [4]) is then used as the initial condition for the model integration. The vectors $\boldsymbol{\Theta}$ and $\mathbf{T}_m$ are N-vectors defined on the model grid $\mathcal{G}_m$; the observation vector $\mathbf{T}_o$ is an M-vector defined on the observation grid $\mathcal{G}_o$.

Usually, $M \ll N$, and $\mathbf{T}_o$ alone cannot provide an adequate representation of $\boldsymbol{\Theta}$. Thus, assimilation is performed resulting in initial conditions on $\mathcal{G}_m$, which are used for the numerical integration of the model equations. We assume there exists a mapping such that $\mathbf{D}(\mathcal{G}_m) = \mathcal{G}_o$. Then $\boldsymbol{\Theta}_o$, the true state of the ocean at the observation grid, can be written as $\boldsymbol{\Theta}_o = \mathbf{D}(\boldsymbol{\Theta})$. Often $\mathbf{D}$ is assumed to be a linear mapping so that

$$\boldsymbol{\Theta}_o = \mathbf{D}\boldsymbol{\Theta}. \tag{2.1}$$

The motivation to perform such data assimilation is the assumption that both the model values and the observational data are unbiased estimators of the true state of the ocean, so that the optimal combination of the two will lead to an estimator of $\boldsymbol{\Theta}$, which has a smaller error variance than either of the two. Under the unbiasedness assumption, we can write the following linear model:

$$\begin{pmatrix} \mathbf{T}_m \\ \mathbf{T}_0 \end{pmatrix} = \begin{pmatrix} \boldsymbol{\Theta} \\ \mathbf{D}\boldsymbol{\Theta} \end{pmatrix} + \begin{pmatrix} \mathbf{e}_m \\ \mathbf{e}_0 \end{pmatrix}, \tag{2.2}$$

where $\mathbf{e}_m$ and $\mathbf{e}_o$ are vectors of zero mean, random errors in the model output, and observations, with Σ_m and Σ_o as their respective covariance matrices. It is reasonable to assume that the errors in $\mathbf{T}_o$ and $\mathbf{T}_m$ are statistically independent. Then the least squares solution to the true state $\boldsymbol{\Theta}$ is obtained by minimizing the quadratic functional

$$\mathcal{Q} = (\mathbf{T}_m - \boldsymbol{\Theta})'\Sigma_m^{-1}(\mathbf{T}_m - \boldsymbol{\Theta}) + (\mathbf{T}_o - \mathbf{D}\boldsymbol{\Theta})'\Sigma_o^{-1}(\mathbf{T}_o - \mathbf{D}\boldsymbol{\Theta}), \tag{2.3}$$

where the prime indicates matrix transposition. It is customary to write Q in terms of corrections to the model values

$$Q = \mathbf{T}_c'\Sigma_m^{-1}\mathbf{T}_c + (\mathbf{D}\mathbf{T}_c - \mathbf{T}_{co})'\Sigma_o^{-1}(\mathbf{D}\mathbf{T}_c - \mathbf{T}_{co}), \tag{2.4}$$

where $\mathbf{T}_c = \mathbf{T}_m - \mathbf{\Theta}$ and $\mathbf{T}_{co} = \mathbf{D}\mathbf{T}_m - \mathbf{T}_o$. Ordinarily, this minimization should be quite simple except for the large dimensions of the model grid and hence that of the covariance matrix Σ_m; thus, the routinely used optimization procedures are rendered inadequate, and more efficient minimization algorithms must be devised.

With a given Σ_m, DR gave an efficient, preconditioned conjugate gradient algorithm for minimizing Q, which avoids computing Σ_m^{-1} (also, see [10]). It iteratively reduces the gradient vector $\mathbf{g}$ to zero; i.e.,

$$\Sigma_m^{-1}\mathbf{T}_c + \mathbf{D}'\Sigma_o^{-1}(\mathbf{D}(\mathbf{T}_c) - \mathbf{T}_{co}) \to 0. \tag{2.5}$$

Preconditioning is supplied by the matrix Σ_m. The iteration steps of the conjugate gradient method are as follows. Let $\mathbf{T}_j$ be the iterative solution to $\mathbf{T}_c$ at iteration j. We start with initialization:

$$\mathbf{T}_1 = \mathbf{0}, \tag{2.6a}$$

$$\mathbf{g}_1 = -\mathbf{D}'\Sigma_o^{-1}\mathbf{T}_0, \tag{2.6b}$$

$$\mathbf{h}_1 = \Sigma_m\mathbf{g}_1, \tag{2.6c}$$

$$\mathbf{d}_0 = \mathbf{e}_0 = \mathbf{0}. \tag{2.6d}$$

Then, the iterations are carried out via equations (2.7a)–(2.7h) given below:

$$\mathbf{d}_n = \mathbf{h}_n + \beta_{n-1}\mathbf{d}_{n-1}, \tag{2.7a}$$

$$\mathbf{e}_n = -\mathbf{g}_n + \beta_{n-1}\mathbf{e}_{n-1}, \tag{2.7b}$$

$$\mathbf{f}_n = \mathbf{e}_n + \mathbf{D}'\Sigma_o^{-1}\mathbf{D}\mathbf{d}_n, \tag{2.7c}$$

$$\alpha_n = \frac{\langle \mathbf{g}_n \mathbf{h}_n \rangle}{\langle \mathbf{d}_n, \mathbf{f}_n \rangle}, \tag{2.7d}$$

$$\mathbf{g}_{n+1} = \mathbf{g}_n + \alpha_n\mathbf{f}_n, \tag{2.7e}$$

$$\mathbf{T}_{n+1} = \mathbf{T}_n + \alpha_n\mathbf{d}_n, \tag{2.7f}$$

$$\mathbf{h}_{n+1} = \Sigma_m\mathbf{g}_{n+1}, \tag{2.7g}$$

$$\beta_{n+1} = \frac{\langle \mathbf{g}_{n+1}, \mathbf{h}_{n+1} \rangle}{\langle \mathbf{g}_n, \mathbf{h}_n \rangle}, \tag{2.7h}$$

where n is the iteration index. These steps are repeated until convergence is achieved. The terms involving Σ_o^{-1} in (2.6b) and (2.7c) are easily computed, since the assumption of statistically independent observation errors yields Σ_o and Σ_o^{-1} as diagonal matrices. The major computation is involved in (2.6c) and (2.7g), where the multiplication $\mathbf{h}_{n+1} = \Sigma_m\mathbf{g}_{n+1}$ is carried out.

For most implemented covariance structures, this operation is quite expensive, more so for repeated applications of the operation. In fact, there is a two-fold problem at this step: one is the multiplication operation that we just mentioned; the second is the practical problem of actual storage of the large-dimensioned computed covariance matrix. Although the computed matrix can be stored on a file and read when needed, the input/output operations can result in excessive demand on computer time, especially in repeated applications. Thus, a fast and accurate algorithm must be developed to handle this matrix multiplication operation.

3. Product-polynomial operator algorithm. The DR algorithm alleviates the above difficulties, to some extent, when the covariance structure of Σ_m is based on a Gaussian function

$$e(r) = a\exp(-r^2/b^2). \tag{3.1}$$

$e(r)$ represents the covariance between two model grid values, r is the distance between the two grid points, and b is the correlation length scale. The parameter a represents the error variance of the model values. With (3.1), the matrix multiplication $\Sigma_m \mathbf{g}$ can be expressed as

$$h(m,n) = \sum_s \sum_t g(s,t) a \exp\left[-\frac{(m-s)^2\delta_1^2 + (n-t)^2\delta_2^2}{b^2} \right]. \tag{3.2}$$

The DR algorithm approximates the $h(m,n)$ array via K applications of the operator $\mathcal{L} = (1 + \frac{\nabla^2}{N})$, where ∇^2 is the discrete Laplacian operator and K is an empirically determined, large integer [11].

A close examination of the DR algorithm reveals two possible improvements. The first is to replace the operator $\mathcal{L}^K$ by some other polynomial in ∇^2, $p(\nabla^2)$ that is a better approximation to (3.2) than $\mathcal{L}^K$. If a more accurate approximation is achieved, then it would be possible to pick the degree of p much lower than K, thus saving many computations when computing $p(\nabla^2)$. As we analyze this possibility, we notice the second improvement: since the quantity we are trying to estimate is a convolution product over a discrete space, it should be possible to give an analysis that exploits this fact directly without using continuous approximations that lead to the application of the operator $\mathcal{L}^K$ [11]. We found that, in this new domain analysis, the Laplacian operator could be replaced by a product of polynomials in simple averaging operators. In fact, the implementation of our observations showed that our algorithm is readily adapted to a much more general covariance structure that is (termed as) separable.

DEFINITION. *A real function of several variables $f(x_1, \dots, x_n)$ is* separable *if it can be expressed as a product of factors that are even functions of a single variable*; *i.e., $f(x_1, \dots, x_n) = \prod f_i(x_i)$ such that $f_i(-s) = f_i(s)$ for all s, and f_i are absolutely integrable.*

A referee has provided us with an intriguing insight into looking at the behavior of the Laplacian operator $\mathcal{L}^K$. Using the analogy of the classical differential equation

$$y' = Ay \quad \text{with} \quad y(0) = y_0,$$

the operator $\mathcal{L}^K$ can be seen as a forward Euler approximation to the associated diffusion equation

$$y' = \nabla^2 y.$$

This suggests that the DR algorithm can be made computationally faster by using better integration methods, e.g., the Adams–Bashforth second-order method, and we fully subscribe to the referee's suggestion. In fact, this suggestion opens up the possibility of any number of modifications to the DR algorithm, depending on the order of the Adams–Bashforth method employed. Although we will improve the computing efficiency, we still would not have the enhancements that the PPO algorithm offers, viz, ease of implementation, generalization to a wider class of covariance functions, parallel computing, and better accuracy in the boundary regions.

We shall derive the PPO algorithm in the general framework of computing convolutions using separable smoothing functions. The Gaussian structure in (3.1) forms a special case.

3.1. Preliminaries and notation. For notational convenience, the details of our development are provided in two-dimensional Cartesian coordinates; the results are easily generalized to higher dimensions. The Fourier transform of g will be denoted by $\hat{g}$, and the convolution operation will be denoted by $*$. We will require the covariance functions to be separable, i.e., in two dimensions, $e(s,t) = f_1(s)f_2(t)$ with $f_k(x) = f_k(-x)$.

Let $\mathcal{Z}^2$ be the group of all ordered pairs (m,n), where m and n are integers. Let $L_1(\mathcal{Z}^2)$ be the set of all sequences $\{a_{mn}\}$ such that $\sum\sum |a_{mn}| < \infty$. We define the matrix operator E on $L_1(\mathcal{Z}^2)$ by $E(g)(m,n) = g * e(m,n)$, where e is a fixed element of $L_1(\mathcal{Z}^2)$. It is easily seen that $g * e \in L_1(\mathcal{Z}^2)$. Since $\widehat{g * e} = \hat{g} \cdot \hat{e}$, this suggests using a typical Fourier analysis on the group. With this setup, we can now derive our algorithm.

The matrix multiplication, analogous to (3.2), is written as

$$h(m,n) = \sum_{s,t} g(s,t)e(m-s,n-t) \tag{3.3a}$$

$$= (g * e)(m,n), \tag{3.3b}$$

where $g * e$ is the convolution of the two, two-dimensional arrays [13]. We will exploit to our advantage the following facts that (1) the covariance functions are separable, which leads to their Fourier transform to be functionally separable and real, and that (2) the matrix multiplication operation $E(g)$ is equivalent to a convolution of two, two-dimensional arrays, so that the Fourier transform of $E(g)$ is a product of the Fourier transforms.

3.2. Derivation of canonical forms. The development of the PPO algorithm is aided by deriving two canonical forms that are functionally similar. One canonical form expresses the two-dimensional convolution $h(m,n)$ in the Fourier domain, and the other represents the resultant of when $g(m,n)$ is operated on by the PPO. A comparison of the two expressions leads to the selection of the polynomials that constitute the PPO.

We would like to show that $h(m,n)$ in (3.3a) can be approximated by successively applying yet-to-be-determined polynomial operators $P_1(D_1)$ and $P_2(D_2)$ where D_k, $k = 1, 2$ are the simple averaging operators given by

$$D_1 g(m,n) = [g(m+1,n) + g(m-1,n)]/2, \tag{3.4a}$$

$$D_2 g(m,n) = [g(m,n+1) + g(m,n-1)]/2. \tag{3.4b}$$

The canonical expressions are given in the following theorems.

THEOREM 1. *Let $\hat{g}(\theta,\phi)$ be the two-dimensional Fourier transform of $g(m,n)$. Then*

$$h(m,n) = \int_0^1 \int_0^1 \hat{g}(\theta,\phi) q_1(\theta) q_2(\phi) e^{2\pi i(m\theta+n\phi)} d\theta d\phi, \tag{3.5}$$

where

$$q_k(\theta) = \sum_s f_k(s) \exp[(-2\pi i s\theta)]. \tag{3.6}$$

THEOREM 2. *Let D_k, $k = 1, 2$ be the averaging operators from* (3.4). *Then*

$$P_2(D_2)P_1(D_1)g(m,n) = \int_0^1 \int_0^1 \hat{g}(\theta,\phi) P_2(\cos 2\pi\phi) P_1(\cos 2\pi\theta) e^{2\pi i(m\theta+n\phi)} d\theta d\phi. \tag{3.7}$$

Note that the right-hand sides of (3.5) and (3.7) are similar functional forms. An examination of the two indicates that we could approximate $h(m,n)$ as

$$h(m,n) \simeq P_2(D_2)P_1(D_1)g(m,n), \tag{3.8}$$

provided $P_1(\cos 2\pi\theta) \simeq q_1(\theta)$ and $P_2(\cos 2\pi\phi) \simeq q_2(\phi)$. How good an approximation (3.8) is depends on how well we can approximate q_k by polynomials P_k over the domain of q_k. First, we prove the two results in (3.5) and (3.7).

Proof of Theorem 1. We start by computing the Fourier transform $\hat{e}$ of the covariance function

$$\hat{e}(\theta, \phi) = \sum\sum e(s, t) \exp(-2\pi is\theta - 2\pi it\phi), \tag{3.9}$$

where θ and ϕ are in the interval [0, 1]. Because $e(s, t)$ is separable, we can write

$$\begin{aligned}\hat{e}(\theta, \phi) &= \left(\sum f_1(s) \exp[(-2\pi is\theta)]\right)\left(\sum f_2(t) \exp[(-2\pi it\phi)]\right)\\ &= q_1(\theta)q_2(\phi),\end{aligned}$$

where $q_k, k = 1, 2$ are defined in (3.6). Thus, $\hat{e}(\theta, \phi)$ "separates" as a product of functions in θ and ϕ. So, from (3.3b), we know $\widehat{E(g)}$ is in the transform space. That is,

$$\widehat{E(g)} = \hat{g} \cdot \hat{e} = \hat{g}q_1q_2. \tag{3.10}$$

But, by the Fourier inversion theorem [7],

$$E(g)(m, n) = \int_0^1 \int_0^1 \widehat{E(g)} e^{+2\pi im\theta} e^{+2\pi in\phi} d\theta d\phi. \tag{3.11}$$

Finally, Theorem 1 is proved when $\widehat{E(g)}$ in (3.11) is replaced by the right-hand side of (3.10). □

Proof of Theorem 2. With $\hat{g}$ as the Fourier transform of g, we can write

$$g(m, n) = \int_0^1 \int_0^1 \hat{g}(\theta, \phi) e^{2\pi i(m\theta+n\phi)} d\theta d\phi. \tag{3.12}$$

Applying operator D_1 from (3.4a) to (3.12), we get

$$\begin{aligned}D_1 g(m, n) &= \int_0^1 \int_0^1 \hat{g}(\theta, \phi) \frac{(e^{2\pi i\theta} + e^{-2\pi i\theta})a}{2} e^{2\pi i(m\theta+n\phi)} d\theta d\phi\\ &= \int_0^1 \int_0^1 \hat{g}(\theta, \phi) \cos(2\pi\theta) e^{2\pi i(m\theta+n\phi)} d\theta d\phi.\end{aligned}$$

We note that every time D_1 operates on g, we get an extra factor of $\cos(2\pi\theta)$ inside the integral. Thus, computing $D_1, D_1^2, \ldots$, it is easy to see that an application of the polynomial operator $P_1(D_1)$ yields

$$P_1(D_1)g(m, n) = \int_0^1 \int_0^1 \hat{g}(\theta, \phi) P_1(\cos 2\pi\theta) e^{2\pi i(m\theta+n\phi)} d\theta d\phi.$$

A similar application of operator polynomial $P_2(D_2)$ yields

$$P_2(D_2)g(m, n) = \int_0^1 \int_0^1 \hat{g}(\theta, \phi) P_2(\cos 2\pi\phi) e^{2\pi i(m\theta+n\phi)} d\theta d\phi.$$

Successive application of the two polynomial operators P_1 and P_2 gives the desired expression (3.7), thus, proving Theorem 2. □

If we approximate $E(g)(m,n)$ by $P_2(D_2)P_1(D_1)(g)(m,n)$, then the error is bounded by

$$\begin{aligned}|(E - P_2(D_2)P_1(D_1))g(m,n)| \le \int_0^1 \int_0^1 |\hat{g}(\theta,\phi)| d\theta d\phi \\ \times \sup |q_1(\theta)q_2(\phi) - P_1(\cos 2\pi\theta)P_2(\cos 2\pi\phi)|,\end{aligned} \tag{3.13}$$

where q_k are given by (3.6). Thus, our goal should be to estimate $q_k(\theta)$ by $P_k(\cos 2\pi\theta)$, $k = 1, 2$.

3.3. Polynomial-operator approximations. By (3.6) we have

$$q_1(\theta) = \sum_s f_1(s) \exp[(-2\pi i s\theta)].$$

Using the assumption that f_1 is an even and real function, then

$$q_1(\theta) = \sum_{s\ge1} f_1(s) 2\cos(2\pi s\theta) + f_1(0).$$

Now, we want to pick a polynomial P_1 so that $P_1(\cos 2\pi\theta)$ is a good approximation to $q_1(\theta)$. To more easily compute this polynomial, we make the change of variables $t = \cos 2\pi\theta$, such that t ranges over $[-1,1]$. Thus, we need to compute $T_k = \cos(2\pi k\theta)$ in terms of t. This is easily achieved using the Chebyshev polynomial recursion formula [1, p. 239]. This result implies there exists a kth degree polynomial T_k such that $T_k(t) = T_k(\cos 2\pi\theta) = \cos(2\pi k\theta)$, and

$$T_{k+1} = 2tT_k(t) - T_{k-1}(t), \quad k = 1, 2, \ldots,$$

where $T_0(t) = 1$ and $T_1(t) = t$. We can compute $q_1(\theta)$ for any value of $t = \cos(2\pi\theta)$ using the recursion to compute $\cos(2\pi s\theta)$ in terms of t and adding up enough terms so that the remainder of the terms will be small. This is possible since $f_i(s)$ is absolutely integrable, which implies $\sum_{s\ge0} |f_i(s)| < \infty$.

Now we want to construct P_1 as an approximation to $q_1(\theta)$ considered as a function of t. The idea here is to compute P_1 as an interpolating polynomial on $[-1,1]$ in t. The question is at which points do we interpolate? We use the expanded Chebyshev points [1, p. 244]. These points are

$$x_i = \left[a + b + (a-b)\cos\left(\frac{2i+1}{2n+2}\pi\right) \Big/ \cos\left(\frac{\pi}{2n+2}\right)\right] \Big/ 2, \quad i = 0, \ldots, n,$$

where $a = -1$, $b = 1$. This choice of points gives a nearly optimal polynomial P_1 such that $\max_{-1\le t\le1} |P_1(t) - q_1(t)|$ is nearly minimized [1, p. 243]. At worst, the error is only about four times as great using these points as it would be if we actually found the optimal polynomial. There are algorithms for computing these optimal polynomials, for example, the second algorithm of Remes and the differential correction algorithm [12, pp. 315–320]. We use the expanded Chebyshev points instead because of their simplicity, and because we have not decided what degree polynomial to use. It will be less of a computational effort to determine the degree using the expanded Chebyshev points than if we were to use these more-complicated algorithms to determine the actual best approximation.

Thus, we let $P_1(t)$ be the interpolating polynomial to $q_1(t) = q_1(\theta)$ at the expanded Chebyshev points. The function $q_1(\theta) = q_1(t)$ is computed to ϵ-tolerance by using the Chebyshev recursion formula, and

$$q_1(\theta) \approx \tilde{q}_1(\theta) = f_1(0) + \sum_{s=1}^{s_0} f_1(s) 2\cos(2\pi s\theta), \tag{3.14}$$

where we pick s_0 such that $2\sum_{s=s_0+1}^{\infty} |f_1(s)| \le Tol_1$, where Tol_1 will be determined below.

3.4. Determination of the degree of P_1. To determine the degree of P_1 requires some computational effort. We first determine an error tolerance Tol_2 (given below). We pick n and find the polynomial of degree n that interpolates $\tilde{q}_1(\theta)$ at the expanded Chebyshev points $x_i, i = 0, \dots, n$. We approximate $\|q_1 - P_1\|$ by $\max_{1\leq i\leq 100} |\tilde{q}_1(\theta_i) - P_1(t_i)|$ where t_i are one hundred equally spaced points on [−1,1]. If $\|\tilde{q}_1 - P_1\| \leq Tol_2$, we stop. Otherwise, we increase the degree. We stop if the tolerance is satisfied or if the degree exceeds some predetermined degree. In our simulations with the Gaussian weighting function, this degree stayed below 10. If the degree becomes too large, the computations become numerically unstable, and the algorithm loses its computational advantage. One should also stop if $||\tilde{q}_1 - P_1||$ starts to increase. If the error goal $||\tilde{q}_1 - P_1|| \leq Tol_2$ is not satisfied then an error message is given, and the degree that minimized $||\tilde{q}_1 - P_1||$ is used. Assuming the error goal is achieved, we have $||q_1 - P_1|| \leq ||q_1 - \tilde{q}_1|| + ||\tilde{q}_1 - P_1|| \leq Tol_1 + Tol_2 = Tol_3$. A similar analysis holds for q_2 and P_2 with $||q_2 - P_2|| \leq Tol_3$.

3.5. Determination of error tolerances. To determine the error tolerances above, we need to keep in mind the goal of making

$$|E(g) - P_2(D_2)P_1(D_1)g(m,n)| \leq \epsilon \quad \forall\ (m,n) \in \mathcal{Z}^2,$$

where ϵ is given. Note ϵ should not be chosen much smaller than the expected error in one's measurements. To determine Tol_1, Tol_2, we need an approximate error estimate

$$\begin{aligned}|E(g) - P_2(D_2)P_1(D_1)g(m,n)| &\leq \int_0^1 \int_0^1 |\hat{g}(\theta,\phi)| d\theta d\phi \|q_1 q_2 - P_1 P_2\| \\ &\leq (M_1 Tol_3 + M_2 Tol_3)||\hat{g}||_1,\end{aligned}$$

where $||\hat{g}||_1 = \int_0^1 \int_0^1 |\hat{g}(\theta,\phi)| d\theta d\phi$, and $M_k = ||q_k||$ (sup-norm). We have also approximated $||P_2|| \approx M_2$. We can estimate $||q_k|| \leq |f_k(0)| + \sum_{s\geq 1} 2|f_k(s)|$, $k = 1, 2$. One can also show that $||\hat{g}||_1 \leq ||g||_1$. Now, if $||g||_1$ is large, the above error estimate is too conservative to be of any use. Instead of an arbitrary g, we choose the special case of $g = \delta_{s,t}$, which is one at (s,t) and zero at all other points. Then $||g||_1 = 1$, and we need to make $Tol_3 \leq \frac{\epsilon}{M_1+M_2}$ and pick $Tol_1 = Tol_2 = \frac{Tol_3}{2}$. This is justified since

$$g(m,n) = \sum_s \sum_t g(s,t)\delta_{s,t}(m,n); \tag{3.15}$$

i.e., $g = \sum_s \sum_t g(s,t)\delta_{s,t}$ and

$$E(g) = \sum_s \sum_t g(s,t)E(\delta_{s,t}). \tag{3.16}$$

So, if we approximate $E(\delta_{s,t})$, we have a good approximation to $E(g)$. In our simulations, the above strategy has worked well. Other error estimates are too pessimistic and, hence, require much more computational effort.

3.6. What to do at the boundary. In practice, data are missing from some finite set. We set this missing data to zero. This is justified since the convolution product contains only nonzero data. The algorithm is applied in one dimension at a time. Each row can be done independently. This means many of these computations can be done in parallel. The nonzero values in the data propagate into the missing data, one unit on each end of a fixed data line, each time we apply D_k. Thus, we need to add temporary memory of size $2n_k$, where n_k = degree of polynomial P_k, $k = 1, 2$. We only need to start the computations at a point once there is a nonzero value on either side of the point.

3.7. Computational effort. When we interpolate q_k at the expanded Chebyshev points $x_0, \dots, x_n$, we obtain a polynomial P_k in Newton form [1, pp. 40–44]

$$P_k(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_n(x - x_0)\dots(x - x_{n-1}).$$

Replacing x by D_k, we need to compute

$$P_k(D_k) = a_0 + a_1(D_k - x_0) + a_2(D_i - x_0)(D_k - x_1) + \cdots + a_n(D_k - x_0)\dots(D_k - x_{n-1}).$$

We evaluate this in Newton nested form, just as we do for a polynomial. Thus, a typical step in this evaluation is to compute $k_1 I + k_2(D - k_3 I)$ where I is the identity operator. One application of such an operator at a grid point requires three multiplications and two additions. Evaluating D requires one more addition and a division by 2 (a shift register, so we will not count this operation). Thus, one step in the nested evaluation is six arithmetic operations—three multiplications and three additions. We do this at each point of the grid. We do this operation n_i times (degree of P_i). Let N be the total number of points on the grid. We can either add $2n_i$ points to the data points or ignore the calculations at the boundary if the degree is small compared with the length of the line. So, we have $6n_i N$ operations so far. After we have computed $P_1(D_1)$, we compute $P_2(D_2)$. So, we have a total of $6N(n_1 + n_2)$ arithmetic operations $(O(N))$. Thus, if the degrees of n_1 and n_2 are fairly small, we have an efficient algorithm. If n_1 and n_2 are large, then there are too many computations, and the numerical stability is a problem. The key question is how large are these degrees? It is thus important not to make unreasonable error requirements.

Additional remarks. This technique fits into the general philosophy found in [14] and references therein. One estimates in an optimal way the response function q_k by selecting coefficients of a filter to best estimate q_k in the infinity (sup-norm) norm. Our algorithm exploits this idea by using a polynomial in the simple averaging operators (filters).

4. Steps for dynamical balance.

In ocean modeling, it is possible to assimilate observational data on any of the fields of temperature, salinity, current velocities, and sea surface elevation. The problem of dynamical imbalance occurs when available observations for assimilation do not include every field. In addition, if the observations provide only a partial spatial coverage and/or if they are too different from the model output, it is possible develop an imbalance. As a counterbalance, "transients" or "gravity waves" are generated, which could lead to model instability.

To avoid generation of the transients, as an additional step, a dynamic initialization is often performed in atmospheric data assimilation [2, 3]. However, using the following conservative steps, such initialization has not been necessary in our ocean data assimilation experiments.

4.1. Assign more weight to model output versus observations. If initially the observations are significantly different from the model output, it is advisable that data be injected into the system without any undue shock. This is achieved by assigning to the model output relatively more weight than the observations. For this, instead of (2.3), we minimize the quadratic functional

$$(4.1) \qquad \mathcal{Q}_\alpha = \alpha(\mathbf{T}_m - \boldsymbol{\Theta})'\Sigma_m^{-1}(\mathbf{T}_m - \boldsymbol{\Theta}) + (\mathbf{T}_o - \mathbf{D}\boldsymbol{\Theta})'\Sigma_o^{-1}(\mathbf{T}_o - \mathbf{D}\boldsymbol{\Theta}).$$

Thus, if we assign 10-to-1 weighting in favor of model output then $\alpha = 10$. Such a weighting will gently nudge the model integration toward the observations. Once the model has adjusted to the observations, the weighting factor α can be modified to an appropriate level.

4.2. Incorporate temporal data window. The effect of data shock can be mitigated by letting the data gradually affect the model dynamics. This is done by implementing a temporal weight factor that gradually allows the data to phase in and out of the window. For instance, DR used a thirty-day triangular window where the weight changes from 0 to 1 and from 1 to 0 as the difference in time between the model and the observation goes from -15 to 0 to $+15$. This temporal window acts as a temporal filter reducing the effect of high frequency data. In fact, due to ocean dynamics such implementation is actually advantageous when comparing the large spatial and time scales with the atmosphere. The length of the temporal window depends on the temporal scales that need to be resolved.

4.3. Increase frequency of assimilation. Imbalance between different fields can be avoided by also increasing the frequency of assimilation. By inserting the temperature data continuously, DR allowed the velocity field to adjust to the density field and noted no adverse effect due to transients except perhaps in the data-sparse equatorial regions. In their assimilation of satellite sea surface elevation data, Ezer and Mellor [6] also found no velocity imbalance with continuous data insertion. However, in their earlier experiments with a larger time between assimilation of sea surface elevation data, Mellor and Ezer [9] had to perform a geostrophic adjustment to the velocity field.

5. Efficiency comparisons. In order to assess accuracy and efficiency of our approximation, we ran three test cases to compute the convolution $E(g)$ given by

$$E(g)(m,n) = \sum_s \sum_t g(s,t) \exp[(-(m-s)^2\delta_1^2 - (n-t)^2\delta_2^2)/b^2], \tag{5.1}$$

with an elliptical Gaussian smoothing function. We implemented two pieces of code in portable FORTRAN and tested the algorithm performance on the CRAY YMP: the first routine, called matrix-mult, performs the actual calculations of the convolution in (5.1) and the second approximates $E(g)$ with the PPO algorithm. We used the first as a basis from which to judge the accuracy of the second, and to compare it with the accuracy of the DR algorithm, which is our de facto standard.

For the first test case, we recall our basis representation (3.15) of the $g(m,n)$ array in terms of the Kronecker delta function. It is clear from (3.16) that to approximate $E(g)$ we must be successful in approximating $E(\delta_{s,t})$ at all (s,t). We choose a uniform grid with a rectangular boundary and approximate $E(\delta_{s,t})$ at three representative locations: the center of the grid, a corner, at the middle point of a side. The three sub-cases will demonstrate the capability of our algorithm at selected, isolated points. In the next test case, we want to test the PPO algorithm performance over a region enclosed by an irregular boundary. Because of our data assimilation interest, the Gulf of Mexico (GOM) boundary was selected for this case, and for an overall picture, we set $g = \sum_s \sum_t \delta_{s,t}$, i.e., $g(s,t) = 1$, at all points of a uniform grid over the region enclosed by the GOM boundary. Finally, as the third case, we test the PPO algorithm performance in an actual data assimilation experiment over the GOM region.

Because the evaluation of the Laplacian ∇^2 at the grid point (i,j) requires information at both $(i \pm 1, j)$ and $(i, j \pm 1)$, the DR algorithm cannot be applied at any point on the boundary. Thus, in choosing the three points for simulations with the Kronecker delta function, we moved three grid spacings inward from the boundary. Also, in order to make a comparison with the DR algorithm, the results from DR had to be normalized to bring its maximum amplitude to the level of the matrix-mult algorithm. In all cases, the grid used was an 86×62 grid, which conformed to the grid dimensions used in the actual data assimilation experiment.

While implementing the PPO algorithm, the number of terms s_0 retained in $\tilde{q}_1$ and $\tilde{q}_2$ in (3.14) are calculated as functions of the grid spacing, a preset tolerance ($Tol = 0.001$),

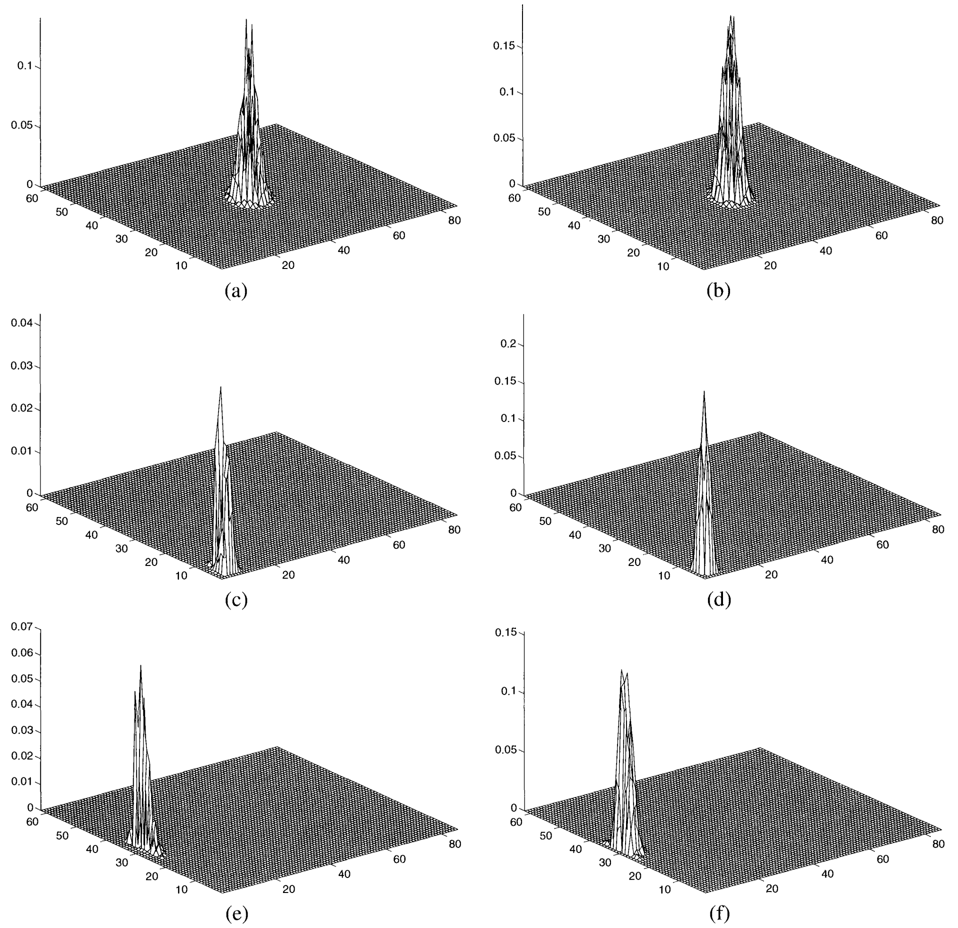


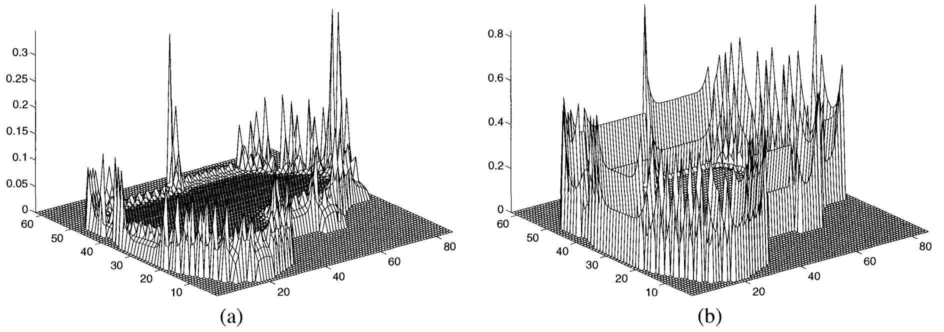
FIG. 1. *Test case* 1. *Gaussian convolution of delta functions at three strategic points—the middle, a corner, and the middle of a side—of a rectangular domain. The results are presented as differences of the PPO and DR algorithms from the matrix-mult algorithm.*

and the scale parameter b as specified for matrix-mult. Explicitly, we set the s_0 such that $|\exp(-s_0^2\delta_k^2/b^2)| \leq Tol$; i.e.,

$$s_0 = \text{int}\,[\sqrt{-\ln(Tol)}/(\delta_i/b)] + 1, \tag{5.2}$$

where int(*) is the greatest integer function. For both $\tilde{q}_k$, (5.2) gave $s_0 = 7$. This led to the Chebyshev polynomials P_k of degree 8 that uniformly approximated the $\tilde{q}_k$'s with an error of 0.000186.

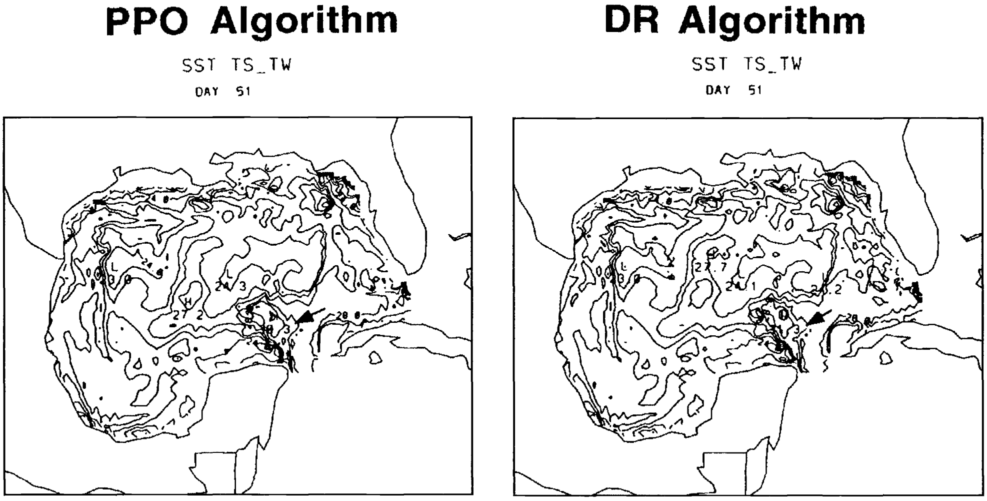
For the first two test cases, the results due to the three algorithms are so similar that it was difficult to tell the differences. Thus, as suggested by a referee, we present the results as differences from the matrix-mult algorithm. Figure 1 shows the results of approximating convolution with the delta function. At the center $(s, t) = (43, 31)$, both the PPO and the DR algorithms exhibit the maximum error around 0.15 (Fig. 1a). However, the PPO algorithm improves at the corner $(s, t) = (3, 3)$ with a maximum error of ~ 0.03, which compares with ~ 0.2 in the DR algorithm (Fig. 1b). The PPO algorithm shows a similar gain in accuracy at the middle of a side of the boundary $(s, t) = (3, 31)$ where PPO versus DR algorithm maximum errors are 0.07 to 0.15 (Fig. 1c).



(a)

(b)

FIG. 2. *Test case 2. Gaussian convolution of $g(s,t) = 1 \; \forall \; s,t$ on an irregular domain provided defined by the Gulf of Mexico topography. The results are presented as differences of the PPO and DR algorithms from the matrix-mult algorithm.*


PPO Algorithm

DR Algorithm

FIG. 3. *Test case 3. Sea-surface temperature from a data assimilation experiment using the PPO and DR algorithms. To the accuracy of the map, the matrix-mult algorithm results (not shown) matched the PPO algorithm. The arrow indicates notable differences between the two algorithms.*

The results from the Kronecker delta simulations indicate that the PPO algorithm should perform well over all points (as was expected from our theoretical analysis), while the DR algorithm may perform poorly over the boundary regions. This fact is borne out in the second test case (Fig. 2). The PPO and DR algorithms perform equally well in the interior of the grid. However, around the boundaries the average PPO errors are around 0.1, while the average errors due to the DR algorithm is around 0.5; the maximum errors are 0.3 for PPO and 0.8 for the DR algorithm.

Finally, to test the performance of the PPO algorithm in a full data assimilation mode, a primitive equation numerical model of the GOM was integrated for forty days without data assimilation, followed by a ten-day integration with assimilation of vertical temperature profile data. The DR algorithm was also tested for comparison using $K = 320$ repeated applications of the Laplacian operator on the same set of data. The model employed a 86×62 grid. The resultant sea-surface temperatures are shown in Fig. 3. To the accuracy of the map, the PPO and matrix-mult algorithms provided identical results; only the PPO algorithm results are shown. While there is a remarkable similarity between the PPO and DR algorithm results, there is noticeable deviation at the location indicated by arrows. The DR algorithm produces a high of 31.3 degrees instead of 30.3 degrees by the PPO (and matrix-mult) algorithm.

In addition, the PPO is quite fast. For a single convolution approximation (matrix multiplication) on a single CRAY YMP processor with "real" data on a 86 × 62 grid, the PPO algorithm takes approximately 0.09 seconds and the DR algorithm takes 2.2 seconds, as compared with 15.4 seconds with matrix-mult. For a comparison of computation times in the full data assimilation mode, the DR algorithm took 6.87 seconds per assimilation step using a single CRAY YMP processor versus .68 seconds with the PPO algorithm. We hasten to add here that the DR algorithm when modified with the Adams–Bashforth second-order method would probably be quite competitive with the PPO algorithm. In the present form, the DR algorithm could be made faster by using values of K somewhat smaller than 320 used in the simulations. However, in our test runs with smaller values, the model integration was unstable.

6. Concluding remarks. We have formulated the problem of data assimilation in terms of a statistical linear model, where the true state of the ocean/atmosphere is estimated by minimizing a quadratic functional. Matrix multiplication $\Sigma_m \mathbf{g}$ is identified to be the major computational bottleneck, where Σ_m is a fixed covariance matrix based on some covariance formulation and $\mathbf{g}$ is a multidimensional, varying vector defined on a specified uniform rectangular grid.

Although, our initial thrust was to improve the efficiency of the DR algorithm based on Gaussian covariance functions, we have made several generalizations:

1. Identify that $\Sigma_m \mathbf{g}$ is a convolution and, thus, express it in the spectral domain (3.5).
2. Identify simple averaging operators (3.4) and show their ability to smooth with a wider class of covariance functions characterized by the separability property; Gaussian covariance functions are an obvious case of separable functions.

The overall effect is that, for computing convolutions, we have come up with a simple, elegant, and efficient algorithm that supports a wider class of covariance structures that may be encountered in the physical sciences. However, the utility of the enhancement to a wider class of covariance functions is yet to be explored. Whereas the Gaussian functions fall into this class quite naturally, the other covariance structures do not appear to be amenable to factorability.

Although, the derivation is carried out in two dimensions, its extension to higher dimensions is obvious. Our Fourier transform pair in one dimension is the integers and the circle group. In higher dimensions, we take the corresponding product of these groups as our Fourier pair.

The algorithm is efficient if one of the functions remains fixed and the convolution is computed many times. This is because we must construct a polynomial fit to a Fourier transform of the fixed factor. If we are able to find a fit within our error tolerance with a low-degree polynomial, then we are able to efficiently and accurately estimate the convolution with our polynomial fit applied to a simple averaging operator.

Acknowledgments. The authors thank the reviewers for their constructive comments. The advice on looking at the DR algorithm in terms of a solution to the diffusion equation was greatly appreciated.

REFERENCES

[1] S. D. Conte and C. deBoor, *Elementary Numerical Analysis*, McGraw-Hill Inc., New York, 1980.

[2] R. Daley, *Variational non-linear normal mode initialization*, Tellus, 30 (1978), pp. 201–218.

[3] ———, *The application of normal mode initialization to an operational forecast model*, Atmosphere Ocean, 17 (1979), pp. 97–124.

[4] R. Daley, *Atmospheric Data Analysis*, Cambridge University Press, New York, 1991.

[5] J. Derber and A. Rosati, *A global data assimilation system*, J. Phys. Ocean., 19 (1989), pp. 1333–1347.

[6] T. Ezer and G. L. Mellor, *Continuous assimilation of GEOSAT altimeter data into a three-dimensional primitive equation model*, J. Phys. Ocean., 24 (1994), pp. 832–847.

[7] G. B. Folland, *Real Analysis*; *Modern Techniques and Their Applications*, John Wiley & Sons, New York, 1984.

[8] R. K. Goodrich, R. M. Passi, and M. Limber, *A large scale Gaussian smoothing algorithm*, Comput. Sci. Statist., 24 (1992), pp. 380–384.

[9] G. L. Mellor and T. Ezer, *A Gulf Stream model and an altimetry assimilation scheme*, J. Geophys. Res., 96 (1991), pp. 8779–8795.

[10] I. M. Navon and D. M. Legler, *Conjugate-gradient methods for large-scale minimization in meteorology*, Mon. Wea. Rev., 115 (1987), pp. 1479–1502.

[11] R. M. Passi, R. K. Goodrich, J. C. Derber, and M. Limber, *An Efficient Data Assimilation Algorithm with a Gaussian Covariance Structure*, COAM Technical Report TR-2/94, Center for Ocean and Atmospheric Modeling, The University of Southern Mississippi, 1993.

[12] A. Ralston and P. Rabinowitz, *A First Course in Numerical Analysis*, McGraw-Hill Inc., New York, 1978.

[13] H. L. Royden, *Real Analysis*, 2nd ed., Macmillan Publishing Co., Inc., New York, 1968.

[14] T. N. Parks and C. S. Burns, *Digital Filter Design*, Wiley Interscience, New York, 1987.

A GENERAL HEURISTIC FOR CHOOSING THE REGULARIZATION PARAMETER IN ILL-POSED PROBLEMS[*]

MARTIN HANKE[†] AND TOOMAS RAUS[‡]

Abstract. For a variety of regularization methods, including Tikhonov regularization, Landweber iteration, ν-method iteration, and the method of conjugate gradients, we develop and illustrate a heuristic for choosing an appropriate regularization parameter. Our choice requires no particular a priori knowledge, since the parameter is determined from computable information only. However, if an estimation for the noise level in the data is at hand, then this can be used as a justification. In contrast to known parameter choice heuristics, a posteriori error estimates for the computed approximations can be given. Numerical examples show that the new parameter choice rules are promising alternatives to known parameter choice rules.

Key words. ill-posed problems, Tikhonov regularization, iterative regularization, conjugate gradients, a posteriori parameter choice

AMS subject classification. 65J10

1. Introduction.

We consider regularization methods for the solution of the ill-posed linear equation

$$Tx = y, \tag{1.1}$$

where T is a bounded operator between Hilbert spaces $\mathcal{X}$ and $\mathcal{Y}$, and y belongs to the range $\mathcal{R}(T)$. We are interested in the solution $x = T^\dagger y$ of (1.1) of minimal norm; here, $T^\dagger$ is the Moore–Penrose generalized inverse of T. Recall that this problem is called ill posed when there lacks a continuous dependency $y \mapsto x$. This is known to be the case if and only if the range of T is nonclosed in $\mathcal{Y}$. As a matter of fact, if only approximate data $\tilde{y} \in \mathcal{Y}$ are given, any straightforward numerical computation of x is unstable (for finite-dimensional discretizations, this will be reflected by large condition numbers of the corresponding matrices).

Ill-posed equations (1.1) arise frequently in the context of *inverse problems*, where it is the aim to recover unknown parameters of a physical system from measurable data. A typical example (although nonlinear) is the determination of the interior conductivity of a medium from measured flux at its boundary; see Engl [3] for several other examples arising in industrial applications.

For a stable numerical approximation of the solution x of (1.1) some regularization technique has to be applied. We refer to the survey [11] for a discussion of different possibilities for incorporating regularization and their algorithmic implementation. Roughly speaking, a regularization method provides a sequence of approximations $\{x_r\}_{r \geq 0}$, with x_0 being some initial guess and $x_r \to T^\dagger y$ as $r \to \infty$, provided exact data $y \in \mathcal{R}(T)$ are available. In general, however, approximate data $\tilde{y}$ do not belong to the domain of $T^\dagger$, in which case $\|x_r\|$ diverges to infinity as $r \to \infty$. Thus, we are left with the important problem of choosing an appropriate regularization parameter r. When r is too big, the numerical implementation will be unstable and x_r will be useless; when r is too small, the approximation x_r is dominated by the initial guess.

Essentially, two types of methods for choosing the regularization parameter r can be distinguished: rules which use information about the noise level $\delta = \|y - \tilde{y}\|$, and rules which do not use such information. The former ones associate each approximation x_r with some positive number η_r (think of the residual norm $\|\tilde{y} - Tx_r\|$, for example), and the regularization

[*] Received by the editors January 26, 1994; accepted for publication (in revised form) March 8, 1995.

[†] Institut für Praktische Mathematik, Universität Karlsruhe, D–76128 Karlsruhe, Germany (hanke@math.uni-karlsruhe.de).

[‡] Department of Mathematics, Tartu University, EE–2400 Tartu, Estonia (yrmitrØ@raud.ut.ee).

parameter $r = r(\delta)$ is chosen such that $\eta_{r(\delta)}$ has about the order of the noise level. Given that the sequence $\{\eta_r\}$ satisfies certain assumptions, convergence of the approximations follows, i.e., $x_{r(\delta)} \to T^\dagger y$ as $\delta \to 0$. Moreover, convergence rates may be derived for the case when the solution $T^\dagger y$ has additional properties. It is clear, however, that the accuracy of these rules depends on precise knowledge of $\|y - \tilde{y}\|$.

The second type of parameter choice rules, on the other hand, seeks to avoid any a priori information, including the noise level. In other words, the chosen regularization parameter $r = r(\tilde{y})$ depends solely on the given data $\tilde{y}$. Typically, these methods are based on a heuristic estimation of the unknown error $\|T^\dagger y - x_r\|$, in one way or the other. Thus we shall speak of *heuristic parameter choice rules* for this type of method below. In practice, such heuristic rules are often the only possibility, because at most only a rough guess of the noise level is usually known. Although these methods often work as well as, or even better than, the rules using information about the noise level (cf. [11] for computational results and comparisons), it was shown by Bakushinskii [1] that no convergence theory as above (i.e., as $\delta \to 0$) can exist for heuristic parameter choice rules. In other words, there always exist "bad" right-hand data $\tilde{y}$ converging to y, for which the corresponding approximations $x_{r(\tilde{y})}$ *fail to converge* to $T^\dagger y$. Because of Bakushinskii's negative result, mathematicians sometimes hesitate to trust heuristic parameter choice rules.

To overcome these concerns we suggest in the following a new methodology for deriving heuristic parameter choice rules for the most important regularization methods. To our knowledge, for each regularization method the resulting algorithm is new, although some similarities to known algorithms may exist. Our approach is based on a very general observation in connection with our experience with so-called order-optimal parameter choice rules (cf. Definition 2.1 below), which preassume knowledge of $\|y - \tilde{y}\|$. In fact, the suggested heuristic rules are always deduced from known and established order-optimal ones. In this way, we are able to derive error bounds for the resulting approximations $x_{r(\tilde{y})}$. A by-product of each new rule is an estimate for $\|y - \tilde{y}\|$. This allows a simple a posteriori check of our parameter choice: if the estimate of the noise level is presumably too small, the regularization parameter should be discarded, and some other parameter choice rule should be used instead.

We emphasize, however, that *no* rule for choosing the regularization parameter in ill-posed problems, and in particular no heuristic rule, should be considered a "black box routine." One can always construct examples where the chosen regularization parameter is far from optimal.

The outline of this paper is as follows. In §2 we motivate our basic idea and present the resulting parameter choice rules for some of the most important regularization methods, namely for Tikhonov regularization, Landweber iteration, and ν-method iteration. In §3, we choose a common framework for the analysis of these regularization methods and derive the theoretical estimates. One of the most important regularization methods is the method of conjugate gradients (CG), because it can be very efficient for large-scale problems. Our algorithm and the theoretical results carry over to this method; see §4. This is probably the most important practical consequence of our general approach, because so far there exist only very few reliable parameter choice rules (i.e., stopping rules) for CG. After a brief comment on some modifications for the discrete setting, we provide some numerical results in §6.

2. Examples. Let us start with a few preliminary remarks. First, we restrict ourselves to problems with $\|T\| \leq 1$. This can be done without loss of generality, since otherwise we simply rescale (1.1). Given perturbed data $\tilde{y}$ with

$$\|y - \tilde{y}\| \leq \delta,$$

the exact solution $T^\dagger y$ can only be approximated within limited accuracy. It is thus a natural approach to ask for approximations that converge to $T^\dagger y$ as quickly as possible as $\delta \to 0$. In this context, we review some results from the literature; cf., e.g., Vainikko and Veretennikov [20] or Louis [14].

DEFINITION 2.1. *Given $\mu > 0$, a parameter choice rule for a regularization method is said to be* order-optimal *(for μ), if the regularization parameter $r = r(y, \tilde{y})$ is chosen in such a way that for some $c > 0$ the error bound*

$$\|T^\dagger y - x_{r(\delta)}\| \leq c\,\delta^{2\mu/(2\mu+1)}\omega^{1/(2\mu+1)} \tag{2.1}$$

holds, whenever $\|y - \tilde{y}\| \leq \delta$ and

$$T^\dagger y - x_0 = (T^*T)^\mu w, \qquad \|w\| = \omega. \tag{2.2}$$

In ill-posed problems T is usually a smoothing operator, hence condition (2.2) corresponds to a certain degree of smoothness of $T^\dagger y - x_0$. It can be shown (cf. [20, 14]) that the exponents on the right-hand side of (2.1) are the best possible under these conditions. Note the loss of information: $T^\dagger y$ can only be approximated within an accuracy of some fractional power of δ. The larger μ is, i.e., the better x_0 approximates the nonsmooth parts of $T^\dagger y$, the better $T^\dagger y$ can be approximated.

The maximum, μ_0 say, of all admissible μ in (2.1) depends on the particular regularization method. We call μ_0 the *qualification* of the method (cf. [20]), and we are only interested in parameter choice rules which are order-optimal for μ_0.

We emphasize that the order-optimal parameter choice rules that we present below are all a posteriori in nature. Besides δ, no further information is required. In particular, we do not need to know whether some condition like (2.2) holds. The point is that *if* such a condition is satisfied *then* we obtain an approximation with order-optimal accuracy, without modifying the algorithm. Let us give some examples.

2.1. Tikhonov regularization. The most well-known regularization technique is Tikhonov regularization; cf. Groetsch [6]. Here, x_r is determined as the minimizer of

$$\|\tilde{y} - Tx\|^2 + \frac{1}{r}\,\|x - x_0\|^2$$

over $x \in \mathcal{X}$. This yields

$$x_r = x_0 + (T^*T + r^{-1}I)^{-1}T^*(\tilde{y} - Tx_0).$$

The qualification of Tikhonov regularization is $\mu_0 = 1$. The *discrepancy principle* is a parameter choice rule, which is order-optimal for $0 < \mu \leq 1/2$; cf. [6]. A parameter choice rule which is order-optimal for all $0 < \mu \leq 1$ has been developed independently by Raus, Engl, and Gfrerer; cf. [17, 5, 4]. It requires that we apply one step of iterative refinement to x_r (cf. [11, Alg. 5.2]),

$$z_r = (T^*T + r^{-1}I)^{-1}T^*(\tilde{y} - Tx_r), \qquad x_r^{\mathrm{II}} = x_r + z_r\,,$$

and that we subsequently determine

$$\eta_r := \langle \tilde{y} - Tx_r^{\mathrm{II}}, \tilde{y} - Tx_r\rangle^{1/2}.$$

Note that x_r^{II} is sometimes called iterated Tikhonov approximation. Given fixed $\tau > 1$, the regularization parameter $r = r(\delta)$ is defined to be the smallest number for which $\eta_r \leq \tau\delta$ holds.

If (2.2) is fulfilled for some $\mu \leq \mu_0$, then the error analysis in [17, 5, 4] shows that for $r > 0$¶

$$\begin{aligned} \|T^\dagger y - x_r\| &\leq c\big((r+1)^{-\mu}\omega + r^{1/2}\delta\big), \\ \eta_r &\leq c\big((r+1)^{-\mu-1/2}\omega + \delta\big). \end{aligned} \tag{2.3}$$

Note that this gives a sequence of bounds corresponding to the continuous range of parameters μ in (2.2). It can further be shown that the upper bounds in (2.3) are the best possible in the sense that there are converse results which specify a sort of one-to-one relation between the exponent μ in (2.2) and the powers $-\mu$ and $-\mu - 1/2$ in (2.3); cf. [6, §3.2] and [16]. In other words, we can expect the sequences $\|T^\dagger - x_r\|$ and η_r to behave asymptotically *like* the right-hand sides of (2.3), where the parameter μ in (2.3) is the supremum of all μ for which (2.2) holds.

Note that the above bound for the error $T^\dagger y - x_r$ attains its minimum for

$$r \sim (\omega/\delta)^{1/(2\mu+1)},$$

which is precisely the same range of r for which the bound for η_r decreases to the order of δ. Further, note that the right-hand side bound for η_r in (2.3) is (up to multiplicative constants) by a factor of $(r+1)^{-1/2}$ smaller than the bound for the error. Because of our discussion on the validity of (2.3) in the previous paragraph, this suggests the following heuristic parameter choice rule.

HEURISTIC PARAMETER CHOICE RULE. *Compute the numbers* $\varphi_r := (r+1)^{1/2}\eta_r, r \geq 0$, *and consider them as approximations of* $\|T^\dagger y - x_r\|$. *Accordingly, determine the (heuristic) regularization parameter* r_* *as a location of the global minimum of* $\{\varphi_r\}$.

We mention that in the special case when $x_0 = 0$, we have

$$\varphi_r = \langle \tilde{y}, r^{-2}(TT^* + r^{-1}I)^{-3}\tilde{y}\rangle^{1/2}.$$

Another heuristic parameter choice rule, namely the *quasi-optimality criterion*, was suggested by Tikhonov and Glasko [18]. They choose the regularization parameter r for which

$$\phi_r = \langle \tilde{y}, r^{-2}(TT^* + r^{-1}I)^{-4}TT^*\tilde{y}\rangle^{1/2} \tag{2.4}$$

attains its minimum. Our theory, however, does not apply to this method. Instead, we refer to Leonov [13].

Other quite successful parameter choice rules for Tikhonov regularization which do not use any information about the noise level are Wahba's generalized cross-validation (cf. [22]) and Hansen's L-curve criterion (cf. [12]).

2.2. Landweber iteration. In the context of ill-posed problems, the method of successive approximations is often called Landweber iteration (cf. [20, 14]). Starting with initial guess x_0, x_{r+1} is obtained from x_r via

$$x_{r+1} = x_r + T^*(\tilde{y} - Tx_r), \qquad r \in \mathbb{N}_0. \tag{2.5}$$

It is known that the iteration (2.5) has an inherent regularizing property. This means that the regularization parameter $r \in \mathbb{N}_0$ is simply the iteration index, and any stopping rule of the iteration renders a parameter choice rule.

¶Here, as below, c denotes a generic constant.

The qualification of Landweber iteration is $\mu_0 = \infty$, and it is known that the discrepancy principle is an order-optimal stopping rule, cf. Vainikko [19]. With this method the iteration is terminated when for the first time

$$\eta_r := \|\tilde{y} - Tx_r\| \leq \tau\delta.$$

Again, $\tau > 1$ is a fixed number.

It is known that the same estimates hold as in (2.3), i.e.,

$$\begin{aligned} \|T^\dagger y - x_r\| &\leq c\big((r+1)^{-\mu}\omega + r^{1/2}\delta\big), \\ \eta_r &\leq c\big((r+1)^{-\mu-1/2}\omega + \delta\big), \end{aligned} \tag{2.6}$$

provided $r \in \mathbb{N}$ and the solution $T^\dagger y$ satisfies (2.2) for some $\mu > 0$. Converse results connecting the validity of the bounds in (2.6) with (2.2) have been established in [7]. With the same argument as above, we therefore suggest the following rule.

HEURISTIC PARAMETER CHOICE RULE. *Compute the numbers*

$$\varphi_r := (r+1)^{1/2}\eta_r = (r+1)^{1/2}\|\tilde{y} - Tx_r\|, \qquad r \in \mathbb{N}_0,$$

and consider them as approximations of $\|T^\dagger y - x_r\|$. *Accordingly, determine the (heuristic) regularization parameter* r_* *as a location of the global minimum of* $\{\varphi_r\}$.

Another order-optimal stopping rule for Landweber iteration has been suggested by Engl and Gfrerer [4]. Instead of the residual they monitor differences between consecutive iterates (see [4] for details). For their method, estimates like (2.6) hold, hence our approach immediately yields another heuristic rule. We are not aware of any further heuristic stopping rules for Landweber iteration.

2.3. ν-methods. A more sophisticated iterative procedure was developed by Brakhage [2]; see also [8, 10]. As opposed to (2.5), the ν-methods are two-step iterative procedures, i.e., x_{r+1} is determined from x_r and x_{r-1} via

$$x_{r+1} = \mu_r x_r + (1-\mu_r)x_{r-1} + \omega_r T^*(\tilde{y} - Tx_r), \qquad r \in \mathbb{N}_0.$$

Here, x_0 is the initial guess and $x_{-1} = 0$. The coefficients μ_r and ω_r can be given explicitly; we refer the reader to the aforementioned papers. The definition of these coefficients depends on a parameter $\nu > 0$ (hence the name ν-method). This parameter also determines the qualification of the method, i.e., we have $\mu_0 = \nu$. Although μ_0 is finite, the ν-methods are much more efficient than Landweber iteration. Roughly speaking, they require only about the square root of iterations compared with Landweber iteration (cf. (2.7) below).

The analysis of the ν-methods is based on their alternative (nonrecursive) definition via Jacobi polynomials. Using this connection, an order-optimal stopping rule (for all $0 < \mu \leq \mu_0$) has been developed only recently; cf. [10]. As for Tikhonov regularization, the discrepancy principle is not order-optimal for the full range $0 < \mu \leq \mu_0$. The order-optimal stopping rule determined in [10] recursively computes a sequence $\{\eta_r\}$ in the course of the iteration and terminates as soon as, for the first time, $\eta_r \leq \tau\delta$ ($\tau > 1$, fixed).

For $r \in \mathbb{N}$ the following bounds are the best possible under the information (2.2) with $0 < \mu \leq \mu_0$:

$$\begin{aligned} \|T^\dagger y - x_r\| &\leq c\big((r+1)^{-2\mu}\omega + r\delta\big), \\ \eta_r &\leq c\big((r+1)^{-2\mu-1}\omega + \delta\big). \end{aligned} \tag{2.7}$$

Note that $r+1$ has taken the role of $(r+1)^{1/2}$ in (2.3) and (2.6). Thus, for the ν-methods, we propose the following modification of the previous stopping rules.

HEURISTIC PARAMETER CHOICE RULE. *Compute the numbers* $\varphi_r := (r+1)\eta_r$, $r \in \mathbb{N}_0$ *and consider them as approximations of* $\|T^\dagger y - x_r\|$. *The computation of* η_r *can be implemented as in* [10, Alg. 4.2]. *Determine the (heuristic) regularization parameter* r_* *as a location of the global minimum of* $\{\varphi_r\}$.

It can be shown that this heuristic stopping rule is asymptotically equivalent to the so-called *update criterion*, which has been suggested previously in [8]. By asymptotically equivalent, we mean that the ratio of φ_r^ϵ as defined in [8, Alg. 3.1] (substituting r for k) and the present φ_r is bounded by positive numbers from above and from below, independent of r, T, and $\tilde{y}$. This means that the theoretical analysis from the following section holds for the update criterion as well (but see also the complementary theoretical results in [8]). We remark that the update criterion results from a completely different motivation.

3. Error bounds. The examples from the foregoing section can be viewed within a well-known spectral theoretical framework; cf., e.g., [6, 20, 14]. We briefly review this general setting, where, for notational convenience, we restrict our attention to the case $x_0 = 0$; the more general case can be handled analogously.

In each of the above examples, $\{x_r\}$ can be expressed via a sequence of functions $\{g_r\}$ as

$$x_r = g_r(T^*T)T^*\tilde{y}. \tag{3.1}$$

Accordingly, we have for the residual $\tilde{y} - Tx_r$ the expression

$$\tilde{y} - Tx_r = h_r(TT^*)\tilde{y} \qquad \text{with} \qquad h_r(\lambda) = 1 - \lambda g_r(\lambda). \tag{3.2}$$

Note that $h_r(0) = 1$.

For example, in Tikhonov regularization we have

$$g_r(\lambda) = \frac{r}{1+r\lambda}, \qquad h_r(\lambda) = \frac{1}{1+r\lambda};$$

Landweber iteration can be defined via (3.1) with

$$g_r(\lambda) = \sum_{k=0}^{r-1}(1-\lambda)^k, \qquad h_r(\lambda) = (1-\lambda)^r;$$

for the ν-method, g_r is such that h_r is a translated and rescaled Jacobi polynomial (see [2, 8] for more details).

We assume throughout that there exists an increasing sequence $\{\rho_r\}$,

$$\begin{aligned} &\rho_0 = 1, \qquad \rho_r \to \infty \quad \text{as } r \to \infty, \\ &\frac{\rho_{r+1}}{\rho_r} \le c \quad \text{for } r \ge 0, \end{aligned} \tag{3.3}$$

such that, uniformly for $\lambda \in [0, 1]$,

$$\begin{aligned} \lambda^{1/2}|g_r(\lambda)| &\le c\rho_r^{1/2}, \\ \lambda^{\mu}|h_r(\lambda)| &\le c\rho_r^{-\mu}, \qquad 0 \le \mu \le \mu_0. \end{aligned} \tag{3.4}$$

Recall that μ_0 is the qualification of the regularization method (3.1). In the examples above, (3.4) is fulfilled with $\rho_r = r + 1$ (Tikhonov, Landweber) and $\rho_r = (r+1)^2$ (ν-methods), respectively.

Since the total error for x_r can be expressed as

$$T^\dagger y - x_r = h_r(T^*T)T^\dagger y + g_r(T^*T)T^*(y - \tilde{y}),$$

one immediately obtains from (3.4) and the spectral theorem the estimates for the error as given in the previous section. Consequently, $1/\rho_r$ can be viewed as *modulus of convergence* of the regularization method.

For the parameter choice rules we also assume that there are functions s_r fulfilling the inequalities

$$\lambda^{\mu} s_r(\lambda) \leq c\rho_r^{-\mu} \qquad \text{for} \quad 0 \leq \mu \leq 2\mu_0 + 1, \ 0 \leq \lambda \leq 1, \tag{3.5}$$

and

$$|h_r(\lambda)|^{2+1/\mu_0} \leq c s_r(\lambda), \qquad 0 \leq \lambda \leq 1. \tag{3.6}$$

If $\mu_0 = \infty$ then we consider $1/\mu_0$ to be zero in (3.6). In many cases, e.g., in Tikhonov regularization and for the Landweber iteration, we will *define* s_r via equality in (3.6). Equality in (3.6) is also the basic assumption for the general theory in [17]. We only know of the ν-methods as a relevant example where the appropriate functions s_r do not satisfy (3.6) with equality.

With the functions s_r we can define an order-optimal parameter choice rule for the regularization method (3.1) via numbers

$$\eta_r = \langle \tilde{y}, s_r(TT^*)\tilde{y}\rangle^{1/2}, \qquad r \geq 0, \tag{3.7}$$

by choosing the smallest regularization parameter $r = r(\|y - \tilde{y}\|)$ for which

$$\eta_r \leq \tau\|y - \tilde{y}\|.$$

Here, τ has to be a fixed number, greater than the supremum over [0, 1] of all functions s_r, $r \geq 0$.

If (2.2) is satisfied for some $0 < \mu \leq \mu_0$, then one easily concludes that

$$\eta_r = \|\big(s_r(TT^*)\big)^{1/2}\tilde{y}\| \leq \|\big(s_r(T^*T)\big)^{1/2}T(T^*T)^{\mu}w\| + \|\big(s_r(TT^*)\big)^{1/2}(\tilde{y} - y)\|;$$

hence (3.5) yields the bound

$$\eta_r \leq c\big(\rho_r^{-\mu-1/2}\omega + \|\tilde{y} - y\|\big). \tag{3.8}$$

Now we state our main result concerning the corresponding heuristic parameter choice rule.

THEOREM 3.1. *Let assumptions* (3.1) *through* (3.6) *be satisfied and assume that* $\|y\| \geq \|y - \tilde{y}\|$, $T^{\dagger}y = (T^*T)^{\mu}w$ *with* $0 < \mu \leq \mu_0$, *and* $\|w\| = \omega$. *Furthermore, let* $r = r_* < \infty$ *be a global minimum of the sequence*

$$\varphi_r := \rho_r^{1/2}\eta_r.$$

Then we have the error bound

$$\|T^{\dagger}y - x_{r_*}\| \leq c\left(1 + \frac{\|\tilde{y} - y\|}{\eta_{r_*}}\right)\delta_*^{2\mu/(2\mu+1)}\omega^{1/(2\mu+1)}, \tag{3.9}$$

where c is a uniform constant and

$$\delta_* = \max\{\eta_{r_*}, \|\tilde{y} - y\|\}. \tag{3.10}$$

Proof. We estimate the two terms in the error decomposition

$$T^{\dagger}y - x_r = h_r(T^*T)T^{\dagger}y + g_r(T^*T)T^*(y - \tilde{y}). \tag{3.11}$$

Consider first the case when $\mu_0 = \infty$. We estimate the first term in (3.11) by means of the moment inequality and assumption (3.4) with $\mu = 0$:

$$\begin{aligned} \|h_r(T^*T)T^\dagger y\| &= \|(T^*T)^\mu h_r(T^*T)w\| \\ &\le \|h_r(T^*T)w\|^{1/(2\mu+1)} \|Th_r(T^*T)(T^*T)^\mu w\|^{2\mu/(2\mu+1)} \\ &\le c\,\omega^{1/(2\mu+1)} \|h_r(TT^*)y\|^{2\mu/(2\mu+1)}. \end{aligned}$$

Using (3.6), this yields

$$\|h_r(T^*T)T^\dagger y\| \le c\,\omega^{1/(2\mu+1)} \langle y, s_r(TT^*)y\rangle^{\mu/(2\mu+1)}. \tag{3.12}$$

Consider next the case when $\mu_0 < \infty$. We define the operator

$$B_r = \big(h_r(T^*T)\big)^{1/\mu_0},$$

and observe from (3.4) that $\{B_r\}$ is uniformly bounded. Applying the moment inequality to $B_r T^*T$, and using the fact that $\mu \le \mu_0$ yields

$$\begin{aligned} \|h_r(T^*T)T^\dagger y\| &= \|(B_r T^*T)^\mu B_r^{\mu_0-\mu} w\| \\ &\le \|B_r^{\mu_0-\mu} w\|^{1/(2\mu+1)} \|T B_r^{\mu_0+1/2} (T^*T)^\mu w\|^{2\mu/(2\mu+1)} \\ &\le c\,\omega^{1/(2\mu+1)} \|\big(h_r(TT^*)\big)^{(2\mu_0+1)/2\mu_0} y\|^{2\mu/(2\mu+1)}. \end{aligned}$$

Thus, from (3.6) it follows that (3.12) holds for $\mu_0 < \infty$ as well. Since

$$\begin{aligned} \|\big(s_r(TT^*)\big)^{1/2} y\| &\le \|\big(s_r(TT^*)\big)^{1/2} \tilde y\| + \|s_r(TT^*)\|^{1/2} \|y - \tilde y\| \\ &\le \eta_r + c\|y - \tilde y\|, \end{aligned}$$

(3.12) with $r = r_*$ implies

$$\|h_{r_*}(T^*T)T^\dagger y\| \le c\,\omega^{1/(2\mu+1)} \big(\eta_{r_*} + c\|y - \tilde y\|\big)^{2\mu/(2\mu+1)}.$$

In other words, using the definition (3.10) of δ_* we have shown:

$$\|h_{r_*}(T^*T)T^\dagger y\| \le c\,\omega^{1/(2\mu+1)} \delta_*^{2\mu/(2\mu+1)}. \tag{3.13}$$

To estimate the second term in (3.11), we first note that

$$\frac{\omega}{\|y - \tilde y\|} \ge \frac{\|y\|}{\|y - \tilde y\|} \ge 1$$

by assumption. Thus, it follows from (3.3) that we can find some number r_0 such that

$$(\omega / \|y - \tilde y\|)^{2/(2\mu+1)} \le \rho_{r_0} \le c\,(\omega / \|y - \tilde y\|)^{2/(2\mu+1)}.$$

Hence, the minimal value φ_{r_*} can be estimated from (3.8) as follows:

$$\varphi_{r_*} \le \varphi_{r_0} \le c\, \|y - \tilde y\|^{2\mu/(2\mu+1)} \omega^{1/(2\mu+1)}. \tag{3.14}$$

Using the first inequality in (3.4) we obtain

$$g_{r_*}(T^*T)T^*(y - \tilde y) \le c\,\rho_{r_*}^{1/2} \|y - \tilde y\| = c\,\varphi_{r_*} \frac{\|y - \tilde y\|}{\eta_{r_*}},$$

where we have used the definition of φ_r for the last equality. Inserting (3.14) and keeping (3.10) in mind, this yields

$$\|g_{r_*}(T^*T)T^*(y-\tilde{y})\| \le c\,\frac{\|y-\tilde{y}\|}{\eta_{r_*}}\,\delta_*^{2\mu/(2\mu+1)}\omega^{1/(2\mu+1)}. \tag{3.15}$$

Finally, (3.11), together with (3.13) and (3.15), verifies the assertion. □

The assumption $\|y\| \ge \|y-\tilde{y}\|$ in the statement of Theorem 3.1 is quite natural. It means that the exact right-hand side is not completely dominated by noise; otherwise, no reasonable reconstruction is possible. Order-optimal parameter choice rules usually return $x_r = 0$ in such a case; since we do not know the magnitude of the perturbation, we cannot do anything alike.

We mention that there is no guarantee that the sequences $\{\varphi_r\}$ as determined above will actually attain a global minimum. It is easy to see that the numbers φ_r are all nonnegative and finite. However, it may happen that $\varphi_r \to 0$ as $r \to \infty$. In this case no global minimum exists. Such an event should be handled with care, and we recommend considering it as a failure of the parameter choice rule; note that one cannot conclude in this case that $\tilde{y}$ belongs to the domain of $T^\dagger$ (cf. [8, Ex. 6.1]).

The situation in which the global minimum is attained for several values of r is not a critical one. Our theory is not affected by this and applies for *all* respective parameters. Nevertheless, we suggest taking r_* as the smallest parameter for which the minimum is attained.

Let us now turn to an interpretation of the bound (3.9). To this end, consider the smallest parameter r_* for which the global minimum of $\{\varphi_r\}$ is attained. Furthermore, let $\delta_* = \eta_{r_*}$ be the resulting estimate for $\|y-\tilde{y}\|$. Note that if we use the order-optimal parameter choice rule with δ_*/τ as an a priori guess for $\|y-\tilde{y}\|$, this method would precisely yield x_{r_*} as a regularized approximation. As long as δ_* is of the order of $\|y-\tilde{y}\|$, it can be seen from Theorem 3.1 that the computed approximations converge to $T^\dagger y$ with order-optimal accuracy. On the other hand, if δ_* does not decrease as quickly as $\tilde{y} \to y$, then the convergence will be suboptimal. More dangerous is the case when δ_* decreases more quickly. In this case, the additional factor $\|y-\tilde{y}\|/\delta_*$ in (3.9) blows up, and the approximations x_{r_*} may diverge. Therefore, the value of δ_* should always be monitored, and the computed approximation should be discarded if δ_* is significantly smaller than the expected noise level.

Stipulating additional conditions, however, we can get rid of the factor $\|y-\tilde{y}\|/\delta_*$ in (3.9). To show this, we denote by Q the orthoprojector in $\mathcal{Y}$ onto the orthogonal complement of the closure of $\mathcal{R}(T)$. To facilitate the argumentation we assume in the sequel for noniterative methods (with regularization parameter $r \in \mathbb{R}_0^+$) that $g_r(\lambda)$ and $s_r(\lambda)$ are continuous with respect to r for every fixed $\lambda \in [0,1]$.

COROLLARY 3.2. *If, in addition to the assumptions of Theorem* 3.1,

$$\|Q(y-\tilde{y})\| \ge \varepsilon\,\|y-\tilde{y}\|$$

for some $\varepsilon > 0$, *then* φ_r *attains a global minimum for some finite* $r_* \ge 0$, *and*

$$\|T^\dagger y - x_{r_*}\| \le \frac{c}{\varepsilon}\,\delta_*^{2\mu/(2\mu+1)}\omega^{1/(2\mu+1)}, \tag{3.16}$$

where δ_* *is as in* (3.10).

Proof. By virtue of (3.6), $s_r(0) \ge c\,h_r(0) = c$, which implies

$$\eta_r = \|\big(s_r(TT^*)\big)^{1/2}\tilde{y}\| \ge \|Q\big(s_r(TT^*)\big)^{1/2}\tilde{y}\| = s_r^{1/2}(0)\,\|Q\tilde{y}\|.$$

Since $Q\tilde{y} = Q(\tilde{y}-y)$ it follows from the additional assumption that

$$\eta_r \ge \varepsilon\sqrt{c}\,\|y-\tilde{y}\|, \tag{3.17}$$

and hence $\varphi_r \to \infty$ as $r \to \infty$. Consequently, there exists a finite r_* for which φ_{r_*} becomes minimal, and the assertion follows from (3.9) and (3.17). □

As will be shown next, if condition (3.18) holds uniformly as $\tilde{y} \to y$, then this is strong enough to prove convergence of $x_{r_*} \to T^\dagger y$.

THEOREM 3.3. *Let $y \in \mathcal{R}(T)$ and $\{y^\delta\}_{\delta>0}$ be a sequence of perturbations of y with $y^\delta \to y$ as $\delta \to 0$. Assume that there exists $\varepsilon > 0$ such that*

$$\|Q(y - y^\delta)\| \geq \varepsilon \|y - y^\delta\| \quad \textit{for all } \delta > 0. \tag{3.18}$$

For each $\delta > 0$, denote by $\{x_r^\delta\}$ the regularized approximation and by $r_*(\delta)$ the regularization parameter defined in Corollary 3.2. Then $x^\delta_{r_*(\delta)}$ converges to $T^\dagger y$ as $\delta \to 0$.

Proof. Let $x = T^\dagger y$ and fix $\delta > 0$. Since $\mu_0 > 0$, it follows from (3.5) that $\rho_r \lambda s_r(\lambda)$ converges to zero as $r \to \infty$, pointwise on $[0, 1]$. Hence, by the Banach–Steinhaus theorem,

$$\|s_r(TT^*)Tx\| = o(\rho_r^{-1/2}), \qquad r \to \infty.$$

Consequently,

$$\begin{aligned} \eta_r &= \|\big(s_r(TT^*)\big)^{1/2} y^\delta\| \leq \|\big(s_r(TT^*)\big)^{1/2} Tx\| + \|\big(s_r(TT^*)\big)^{1/2}(y^\delta - y)\| \\ &\leq o(\rho_r^{-1/2}) + c\,\|y - y^\delta\|, \end{aligned}$$

and

$$\varphi_r = \rho_r^{1/2}\eta_r \leq o(1) + c\rho_r^{1/2}\|y - y^\delta\|, \qquad r \to \infty.$$

Choose $r = r(\delta)$ such that $r(\delta) \to \infty$ and $\rho_{r(\delta)}^{1/2}\|y - y^\delta\| \to 0$ as $\delta \to 0$: this implies

$$\varphi_{r_*(\delta)} \leq \varphi_{r(\delta)} \longrightarrow 0, \qquad \delta \to 0.$$

Because of (3.18), inequality (3.17) with $\tilde{y} = y^\delta$ follows as in the proof of Corollary 3.2, hence,

$$\rho_{r_*(\delta)}^{1/2}\|y - y^\delta\| \leq \frac{1}{\varepsilon\sqrt{c}}\rho_{r_*(\delta)}^{1/2}\eta_{r_*(\delta)} = \frac{1}{\varepsilon\sqrt{c}}\varphi_{r_*(\delta)} = o(1), \qquad \delta \to 0. \tag{3.19}$$

Since $\rho_r \geq 1$, $\varphi_{r_*(\delta)}$ can go to zero only if $\eta_{r_*(\delta)}$ goes to zero as well. This leads to two cases which have to be distinguished. Assume first that a subsequence of $\{r_*(\delta)\}$ converges to some $r_\infty < \infty$ as $\delta \to 0$: then $\eta_{r_\infty} = 0$ and (3.6) implies $Tx_{r_\infty} = y$; by continuity, the corresponding approximations $x_{r_*(\delta)}$ converge to $x_{r_\infty} = T^\dagger y$. Assume next that a subsequence of $\{r_*(\delta)\}$ goes to infinity as $\delta \to 0$: this, together with (3.19) implies that both terms in the error decomposition (3.11) go to zero; hence, $x_{r_*(\delta)} \to T^\dagger y$ as $\delta \to 0$. □

To prove stronger results, for instance convergence rates for $x_{r_*} \to T^\dagger y$, we need to find better bounds than (3.13) for the approximation error. Such bounds may be obtained, for example, if r_* can be estimated from below. This, however, is an extremely difficult problem, very much related to the aforementioned converse results.

4. The conjugate gradient method. One of the most efficient regularization methods is the conjugate gradient method (CG) applied to the normal equation $T^*Tx = T^*y$ (cf. [14, 9]). Like the other iterative methods from §2 it has an inherent regularizing property; that is, the iteration count $r \in \mathbb{N}_0$ can be seen as the regularization parameter. Although the CG-iterates can be expressed as in (3.1), they cannot be analyzed with the techniques from the previous section, since the functions g_r to be used in (3.1) depend on $\tilde{y}$.

Nemirovskii [15] has shown that the qualification of CG is $\mu_0 = \infty$, and that the discrepancy principle is an order-optimal stopping rule. Furthermore, for $r > 0$, Nemirovskii obtained (more or less explicitly) the following bounds, provided (2.2) is fulfilled and $\|y - \tilde{y}\| \le \delta$:

$$\begin{aligned} \|T^\dagger y - x_r\| &\le c(\rho_r^{-\mu}\omega + \rho_r^{1/2}\delta), \\ \eta_r &\le c(\rho_r^{-\mu-1/2}\omega + \delta). \end{aligned} \tag{4.1}$$

Here, as in Landweber's method, $\eta_r = \|\tilde{y} - Tx_r\|$; ρ_r in (4.1) is defined as

$$\rho_0 = 1, \qquad \rho_r = g_r(0), \quad r \in \mathbb{N}.$$

Note that in all examples from §2 we have $\rho_r \approx g_r(0)$ for $r \in \mathbb{N}$, the ratio of the two numbers going to 1 as $r \to \infty$.

Our considerations motivate the following stopping rule for CG.

HEURISTIC PARAMETER CHOICE RULE. *Compute the numbers*

$$\varphi_r := \rho_r^{1/2}\eta_r = \max\{1, g_r(0)\}^{1/2}\|\tilde{y} - Tx_r\|, \qquad r \in \mathbb{N}_0,$$

and consider them as approximations of $\|T^\dagger y - x_r\|$. *Accordingly, determine the (heuristic) regularization parameter* r_* *as a location of the global minimum of* $\{\varphi_r\}$.

We emphasize that $g_r(0)$ is easily computed recursively in the course of the iteration (cf. display (2.11) in [9]), and hence the above stopping rule is efficiently implementable. Theorems 3.1 and 3.3 and Corollary 3.2 hold accordingly. For the proofs we refer to [9]. They are considerably more complicated and rely on techniques developed by Nemirovskii in [15]. We finally mention that the idea of computing $g_r(0)$ for diagnostic purposes has previously been suggested in [14][1]. So far, there is no other heuristic stopping rule for CG with a theoretical foundation; cf. [11].

5. The discrete setting. For a numerical implementation, (1.1) has to be discretized to obtain a finite-dimensional matrix equation

$$Ax = b. \tag{5.1}$$

For this discrete problem, however, we encounter an additional difficulty. For example, if the null space of A is trivial then the normal equations system for (5.1) always has a solution, whatever data $\tilde{b}$ are used. Denote this solution by $\tilde{x}$. Then, the computed numbers φ_r^{comp} defined via

$$\eta_r^{\text{comp}} := \big(\tilde{b}^* s_r(AA^*)\tilde{b}\big)^{1/2}, \qquad \varphi_r^{\text{comp}} := \rho_r^{1/2}\eta_r^{\text{comp}},$$

satisfy

$$\varphi_r^{\text{comp}} = \rho_r^{1/2}\big((A\tilde{x})^* s_r(AA^*)A\tilde{x}\big)^{1/2} = \|\big(\rho_r s_r(A^*A)A^*A\big)^{1/2}\tilde{x}\|.$$

It is an immediate consequence of (3.5) and the Banach–Steinhaus theorem that

$$\big(\rho_r s_r(A^*A)A^*A\big)^{1/2}\tilde{x} \to 0 \qquad \text{as } r \to \infty.$$

Therefore, the computed quantities φ_r^{comp} always converge to zero as $r \to \infty$, i.e., the global infimum of $\{\varphi_r^{\text{comp}}\}$ equals zero.

[1] Be aware of the misprint for $g_r(0)$ on p. 127 of [14].

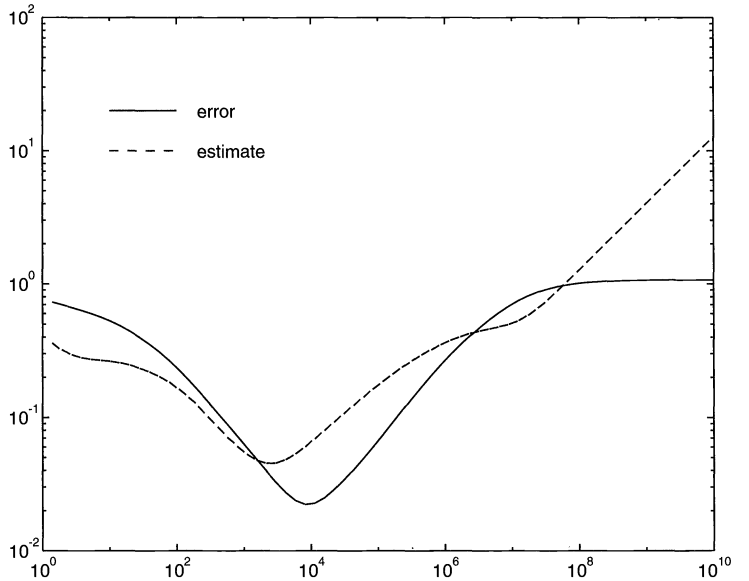


FIG. 5.1. *Error history and estimates $\{\varphi_r\}$ versus r for Tikhonov regularization.*

We emphasize that this phenomenon is only due to the discretization process: the computed numbers φ_r^{comp} and the numbers φ_r for the continuous problem (1.1) are no longer related for $r > r_\infty$, say. In fact, in our computations we have typically observed a significant increase of φ_r^{comp} as r becomes large, before φ_r^{comp} eventually starts to decay towards zero. To satisfy formal needs, we might nevertheless modify our parameter choice rule by first defining r_∞ as the parameter r where φ_r^{comp} eventually starts to decrease, and by choosing r_* as the location of the global minimum of $\{\varphi_r^{\text{comp}}\}$ in $[0, r_\infty]$. However, in practical computations, the parameter r is rarely made so large that one would encounter this phenomenon; instead, the location of the "global minimum" of φ_r^{comp} is usually obvious. To illustrate this claim we show a typical plot of the error history $\|T^\dagger y - x_r\|$ and its approximations φ_r in Figure 5.1. This plot shows the results for a sample run of Tikhonov regularization corresponding to the problem described in §6.2 below with 0.1% noise. The plot exhibits a distinct minimum of the comparison sequence $\{\varphi_r\}$, and a "hump" as r goes to infinity. The magnitude of the regularization parameter r_* is the same as for the optimal regularization parameter; the error $\|T^\dagger y - x_{r_*}\|$ is by about a factor of 1.8 worse than the best possible error.

Another point that we would like to mention is the behavior of our method when we increase the level of discretization. Since the theory developed in §3 is independent of the dimension n of the chosen Hilbert space, we expect similar outputs when the discrete problems approach the continuous problem, i.e., when $n \to \infty$. In fact, in our numerical experiments the sequences $\{\varphi_r\}$ did not change much when we increased n, but kept the relative data error at a constant level. As a consequence, the chosen regularization parameter r_* stagnated at a certain level. Moreover, the estimates η_{r_*} for $\|y - \tilde{y}\|$ in Theorem 3.1 approached a constant value as $n \to \infty$, comparable to the true data error. In view of this, the error bound (3.9) of Theorem 3.1 asserts order-optimal accuracy, which is what we want to get.

We mention, however, that with increasing n the best possible regularization parameter typically goes to infinity, and the error of the corresponding approximation goes to zero. This is a phenomenon which cannot be explained by the "deterministic" continuous setting that we have used for our paper, but is in agreement with the statistical approach to regularization methods advocated by Wahba [22] and others. The method of generalized cross-validation for estimating the regularization parameter is based on this theory, and captures the expected behavior of the optimal regularization parameter as $n \to \infty$. There is no hint, however, that generalized cross-validation can realize error bounds like (3.9) or even order-optimal accuracy.

Just recently, Vogel [21] has shown that the regularization parameter chosen by the L-curve criterion also stagnates when $n \to \infty$. It would be interesting to find a method for which deterministic error bounds like (3.9) hold, and which give the "correct" (i.e., expected) behavior of the regularization parameter as $n \to \infty$. We plan to come back to this problem in some future work.

6. Numerical results. We now summarize some of our numerical experiments with two model problems. For each problem, perturbed data are generated by adding 1% and 0.1% noise, respectively (i.e., if b and $\tilde{b}$ denote the exact and the perturbed right-hand sides then $\|b - \tilde{b}\| / \|b\| = 0.01$ or 0.001). As usual, the noise is a random vector, normally distributed with zero mean. For each of the two noise levels we have applied Tikhonov regularization, the ν-method with $\nu = 1$, and CG to (the same) 20 noisy copies of the right-hand side, respectively. In all the examples, $x_0 = 0$ is the initial guess.

6.1. Model problem 1: A compact operator equation. Our first problem is taken from [8]. It is a first-kind integral equation

$$(Tx)(s) = \int_0^1 k(s,t)x(t)\,dt = y(s), \qquad 0 \le s \le 1,$$

where

$$k(s,t) = \begin{cases} \pi^2 s(1-t), & 0 \le s \le t \le 1, \\ \pi^2 t(1-s), & 0 \le t < s \le 1 \end{cases}$$

is the Green's function for the differential operator $Lx = -x''/\pi^2$ with boundary conditions $x(0) = x(1) = 0$. The integral equation is discretized by collocation with piecewise linear splines and 51 equidistant collocation points $s_j = j/50$, $j = 0, \dots, 50$. As in [8] two different right-hand sides are considered, namely

$$\begin{aligned} y_1(s) &= (7s^6 - 18s^5 + 15s^4 - 5s^3 + s)\pi^2/3, \\ y_2(s) &= \left(-7s^8 + 24s^7 - 28s^6 + 14s^5 - \frac{28}{3}s^3 + \frac{19}{3}s\right)\pi^2/14. \end{aligned}$$

The two right-hand sides differ in the admissible range for the parameter μ in (2.2): for y_1, (2.2) is satisfied for $0 < \mu < 5/8$; for y_2 the range is $0 < \mu < 9/8$.

6.2. Results for y_1. In the sequel, we describe the results with $y = y_1$. We restrict our attention to a comparison of the order-optimal and heuristic parameter choice rules, and to their accuracy as compared to the best possible regularization parameter. For the order-optimal rules we have to fix a parameter τ, which in all cases must be greater than 1; cf. §2. We took $\tau = 1.1$, since our experience indicates that this is a reasonable choice for many applications. In Figure 6.1 the relative errors of the regularization methods are shown. The two plots correspond to the two different noise levels. Each plot contains three columns: the left-hand column presents the results for Tikhonov regularization, the middle column corresponds to the ν-method, and the right-hand side column contains the results for CG. In every column, for each of the 20 experiments (corresponding to the 20 different perturbations of the right-hand side) the relative errors are drawn for the optimal regularization parameter (solid line), the order-optimal parameter choice rule (dotted line), and the corresponding heuristic parameter

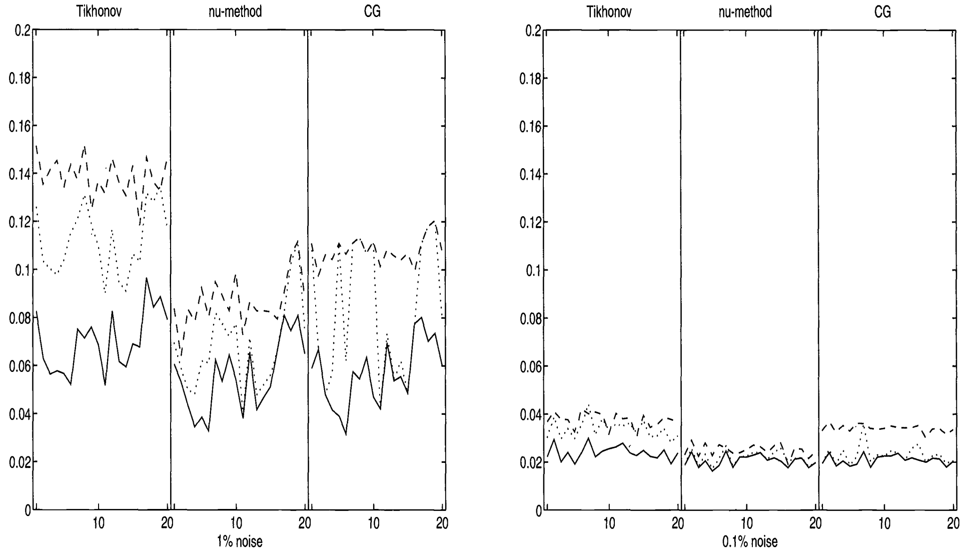


FIG. 6.1. *Model problem* 1, $y = y_1$*: relative errors for* 1% *noise* (*left*) *and* 0.1% *noise* (*right*).

TABLE 6.1
Model problem 1, $y = y_1$*: average iteration counts.*

	ν-method			Conjugate gradients		
Noise level	Opt.	Ord.	Heu.	Opt.	Ord.	Heu.
1%	48.0	33.9	27.0	4.5	3.6	3.0
0.1%	111.9	98.9	82.0	7.5	6.0	5.0

choice rule (dashed line). The actual accuracy is not so important; what matters is the relative accuracy of the competing rules.

Table 6.1 complements the figure by providing the average iteration count of the two iterative schemes corresponding to the optimal iteration index (Opt.), to the order-optimal stopping rule (Ord.), and to the heuristic stopping rule (Heu.).

As can be seen from Figure 6.1, the order-optimal stopping rules provide better approximations than the heuristic rules. The stability of the reconstructions is comparatively robust, i.e., the accuracy is quite independent of the actual noise sample. Although less information is used, the heuristic parameter choice rules perform not much worse than the deterministic rules. In fact, the error of the heuristic rules is never more than twice the error of the corresponding order-optimal rule, except for the case of CG with 1% noise, where differences up to a factor of 2.5 occur. However, it should be remarked that CG requires only about four to five iterations to reach optimal accuracy in the 1% noise case, and the stopping indices of the two stopping rules under consideration never differ by more than 1 (cf. Table 6.1); it seems unrealistic to expect a better coincidence of the stopping indices.

In view of Theorem 3.1 it is also instructive to consider the ratio $\eta_{r_*} / \|y - y^\delta\|$; cf. (3.9). In all experiments this ratio is between 0.89 and 1.41. With 0.1% noise, the ratio never drops below 0.96. In other words, the heuristic rules provide very reliable approximations.

6.3. Results for y_2. Figure 6.2 and Table 6.2 illustrate the numerical results for the first model problem with right-hand side $y = y_2$. Since y_2 satisfies (2.2) for larger values of μ as compared to y_1, the iterative methods require less iterations to reach optimal accuracy.

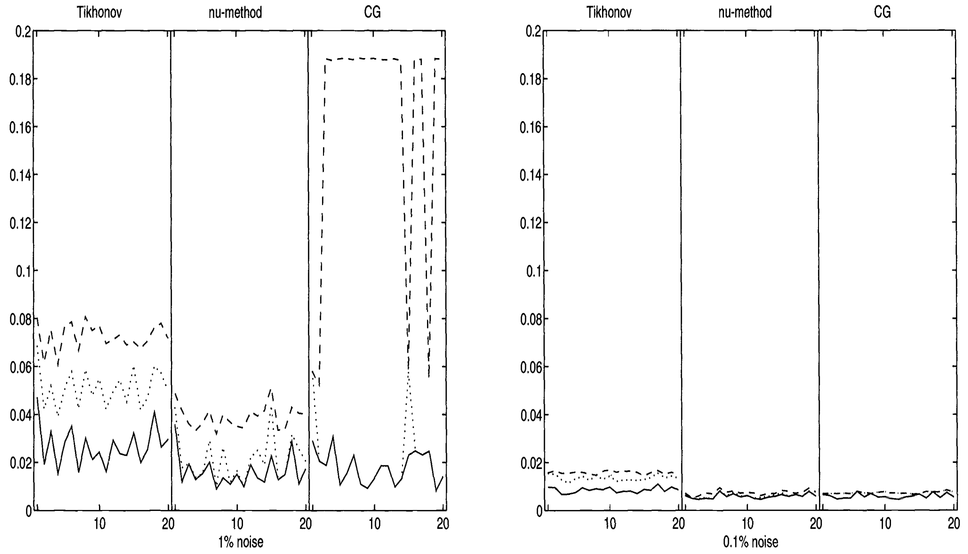


FIG. 6.2. *Model problem* 1, $y = y_2$: *relative errors for* 1% *noise* (*left*) *and* 0.1% *noise* (*right*).

TABLE 6.2
Model problem 1, $y = y_2$: *average iteration counts.*

	ν-method			Conjugate gradients		
Noise level	Opt.	Ord.	Heu.	Opt.	Ord.	Heu.
1%	16.8	13.9	10.0	3.0	2.9	1.2
0.1%	31.8	33.8	29.1	3.9	3.0	3.0

This is evident from Table 6.1, and it particularly applies to CG. In fact, $r = 3$ is throughout the optimal stopping index for CG in the 1% noise case. While this is essentially matched by the order-optimal rule (except for two experiments where stopping index is $r = 2$), the heuristic rule returns $r_* = 2$ in four and $r_* = 1$ in sixteen experiments, respectively. This is the reason for the comparatively large gap in the accuracy between the two stopping rules in the left-hand figure. For 0.1% noise, both stopping rules terminate CG after three iterations whatever perturbation is used, whereas the optimal stopping index is almost always $r = 4$.

As above, the a posteriori estimates η_{r_*} for the perturbation error in the data turn out to be very good approximations, except for the outlying CG approximations in the 1% noise case where $r_* = 1$. For these latter approximations η_{r_*} goes up to almost five times the actual noise magnitude. Nevertheless, as has been mentioned earlier, it would be more dramatic if the ratio dropped significantly *below* 1.

6.4. Model problem 2: A convolution over $\mathbb{R}^2$. The second model problem is a two-dimensional deconvolution taken from [11, §8.2]. The corresponding integral operator T has continuous spectrum instead of a sequence of isolated singular values as in the previous problem. We include this problem to see whether these spectral properties affect the performance of the parameter choice rules. For details on the problem and the (simple collocation) discretization we refer to [11]. Figure 6.3 and Table 6.3 provide the results of the numerical experiments (the properties of the additive noise are the same as before).

Note that the iterative methods require many more iterations; this is due to the fact that the solution has very sharp contours; the actual values of μ for which (2.2) holds are not known.

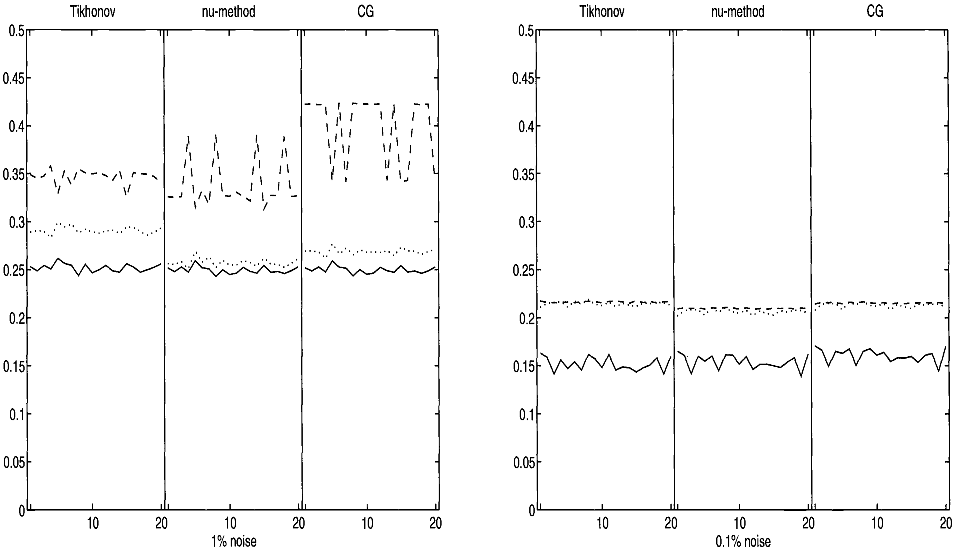


FIG. 6.3. *Model problem* 2*: relative errors for* 1% *noise* (*left*) *and* 0.1% *noise* (*right*).

TABLE 6.3
Model problem 2*: average iteration counts.*

	ν-method			Conjugate gradients		
Noise level	Opt.	Ord.	Heu.	Opt.	Ord.	Heu.
1%	68.0	45.4	17.6	30.8	18.2	6.5
0.1%	871.1	177.2	151.2	209.8	52.2	47.4

Furthermore, CG needs more iterations because the spectrum of T is a continuum; see the discussion in [11].

From a qualitative point of view, the robustness of the parameter choice rules does not change at all. As before, the errors of the regularized approximations obtained with the heuristic parameter choice rules are at most by a factor of 1.6 worse than those of the order-optimal rules; for the reduced noise level, the ratio between the corresponding errors decreases even down to at most 1.04.

On the other hand, the a posteriori estimates for the 1% noise level are larger, up to 4.5 times the actual value. For 0.1% noise, however, all estimates are between 1 and 1.5 times the actual magnitude.

7. Concluding remarks. Heuristic parameter choice rules have found renewed interest in recent works on the regularization of ill-posed problems. The major drawback of many of the suggested rules is the lack of a thorough theoretical analysis. With this in mind, the parameter choice rules presented in this paper are especially attractive, since they are based on the established theoretical framework developed for order-optimal parameter choice rules. As a consequence, we obtain a posteriori bounds for the actual error of the regularized approximation; cf. Theorem 3.1.

We conclude from the numerical experiments that the actual performance of the new parameter choice rules is quite promising, except for the conjugate gradient method, where the results are somewhat worse in the case of 1% noise. With even more noise the heuristic parameter choice rules tend to a too-conservative choice of the parameter; however, it was always possible to detect this from the computed estimate for $\|y - \tilde{y}\|$. Note that with significantly more noise than 1%, sound reconstructions are almost impossible, anyway.

It must again be emphasized that the proposed rules should not be understood as black box routines, and the presented numerical results can only indicate the limitations of these rules. Other examples naturally will meet other limitations. It is our philosophy and our intention to suggest these rules primarily for diagnostic purposes; the use of a variety of different parameter choice rules should always be the ultimate goal.

Acknowledgments. We are grateful to Robert Plato for pointing out a simplified proof of Corollary 3.2, and to Curtis Vogel and an anonymous referee for useful remarks concerning the presentation. The second author greatly appreciates travel support by the Christian-Doppler-Society, Austria.

REFERENCES

[1] A. B. Bakushinskii, *Remarks on choosing a regularization parameter using the quasi-optimality and ratio criterion*, USSR Comput. Math. and Math. Phys., 24, 4 (1984), pp. 181–182.

[2] H. Brakhage, *On ill-posed problems and the method of conjugate gradients*, in Inverse and Ill-Posed Problems, H. W. Engl and C. W. Groetsch, eds., Academic Press, Boston, 1987, pp. 165–175.

[3] H. W. Engl, *Regularization methods for the stable solution of inverse problems*, Surveys Math. Indust., 3 (1993), pp. 71–143.

[4] H. W. Engl and H. Gfrerer, *A posteriori parameter choice for generalized regularization methods for solving linear ill-posed problems*, Appl. Numer. Math., 4 (1988), pp. 395–417.

[5] H. Gfrerer, *An a posteriori parameter choice for ordinary and iterated Tikhonov regularization of ill-posed problems leading to optimal convergence rates*, Math. Comp., 49 (1987), pp. 507–522.

[6] C. W. Groetsch, *The Theory of Tikhonov Regularization for Fredholm Equations of the First Kind*, Pitman, Boston, 1984.

[7] M. Hanke, *Accelerated Landweber iterations for the solution of ill-posed equations*, Numer. Math., 60 (1991), pp. 341–373.

[8] ———, *An ϵ-free a posteriori stopping rule for certain iterative regularization methods*, SIAM J. Numer. Anal., 30 (1993), pp. 1208–1228.

[9] ———, *Conjugate Gradient Type Methods for Ill-Posed Problems*, Pitman Res. Notes Math., Longman, Harlow, 1995.

[10] M. Hanke and H. W. Engl, *An optimal stopping rule for the ν-method for solving ill-posed problems, using Christoffel functions*, J. Approx. Theory, 79 (1994), pp. 89–108.

[11] M. Hanke and P. C. Hansen, *Regularization methods for large-scale problems*, Surveys Math. Indust., 3 (1993), pp. 253–315.

[12] P. C. Hansen, *Analysis of discrete ill-posed problems by means of the L-curve*, SIAM Review, 34 (1992), pp. 561–580.

[13] A. S. Leonov, *On the choice of regularization parameters by means of the quasi-optimality and ratio criteria*, Soviet Math. Dokl., 19 (1978), pp. 537–540.

[14] A. K. Louis, *Inverse und schlecht gestellte Probleme*, B.G. Teubner, Stuttgart, 1989.

[15] A. S. Nemirovskii, *The regularization properties of the adjoint gradient method in ill-posed problems*, USSR Comput. Math. and Math. Phys., 26, 2 (1986), pp. 7–16.

[16] A. Neubauer, *On converse and saturation results for Tikhonov regularization of linear ill-posed problems*, SIAM J. Numer. Anal., to appear.

[17] T. Raus, *The principle of the residual in the solution of ill-posed problems with nonselfadjoint operator*, Uchen. Zap. Tartu Gos. Univ., 715 (1985), pp. 12–20. (In Russian.)

[18] A. N. Tikhonov and V. B. Glasko, *Use of the regularization method in non-linear problems*, USSR Comput. Math. and Math. Phys., 5, 3 (1965), pp. 93–107.

[19] G. M. Vainikko, *Error estimates of the successive approximation method for ill-posed problems*, Automat. Remote Control, 40 (1980), pp. 356–363.

[20] G. M. Vainikko and A. Y. Veretennikov, *Iteration Procedures in Ill-Posed Problems*, Nauka, Moscow, 1986. (In Russian.)

[21] C. Vogel, *Non-convergence of the L-curve regularization parameter selection method*, Inverse Problems, to appear.

[22] G. Wahba, *Spline Models for Observational Data*, Society for Industrial and Applied Mathematics, Philadelphia, 1990.

SIAM J. SCI. COMPUT.
Vol. 17, No. 4, pp. 973–992, July 1996

012

BAYESIAN-VALIDATED SURROGATES FOR NOISY COMPUTER SIMULATIONS; APPLICATION TO RANDOM MEDIA*

SERHAT YEŞİLYURT†, CHAHID K. GHADDAR‡, MANUEL E. CRUZ‡, AND ANTHONY T. PATERA‡

Abstract. Many numerical simulations remain too resource-intensive to be directly incorporated into engineering design and optimization efforts. An attractive alternative to direct insertion considers models for computational systems: the expensive simulation is evoked only to construct and validate a simplified input–output model; this simplified input–output model then serves as a simulation *surrogate* in ensuing design and optimization studies. We present here a nonparametric statistical procedure for the validation and optimization-purposive application of simplified (static) input–output models for *noisy* computer simulations.

For the case of unbounded symmetric measurement noise, we show that, with probability greater than $1 - \varepsilon_2$, the difference between the *noise-free* simulation output and the proposed surrogate is bounded over more than $1 - \varepsilon_1$ of the specified input domain by U, the largest discrepancy observed between the *noisy* simulation output and the proposed surrogate in a validation sample of size $N = \ln\varepsilon_2 / \ln(1 - \varepsilon_1/2)$. Several a priori and a posteriori density-independent and density-dependent bounds are presented for the expected noise contribution to the model prediction error estimator, U; for the important case of a normal parent, bias-variance balance requires that the standard deviation of the noise decrease no more slowly than $(2\ln N)^{-1/2}$ as $N \to \infty$.

As an example, we consider the selection of the inclusion concentration of a random fibrous composite to obtain a desired effective thermal conductivity: a parallel Monte-Carlo finite-element simulation is polled to validate a microstructure-independent effective-conductivity upper bound; this upper bound then serves as a simulation surrogate in subsequent discrimination exercises. A validation-based nonparametric a posteriori error framework is evoked to quantify the effects of surrogate-for-simulation substitution on system predictability and optimality.

Key words. design, optimization, simulation surrogates, nonparametric validation, parallel computing, PAC-learning algorithms, order statistics, Monte-Carlo methods, random media

AMS subject classifications. 80A20, 65N30, 65Y05, 65C05, 65C20, 62G07, 62G15, 62G30, 65K99

1. Introduction.

Computer simulation plays an increasingly important role in engineering design and optimization. Two alternative approaches to the incorporation of simulation into engineering synthesis are *direct insertion* and *simulation surrogates*.

In *direct insertion*, the simulation is directly evoked by the parent mathematical programming procedure. Direct insertion permits accurate system-performance prediction and great flexibility in design-space definition and dimension. However, direct insertion is often not viable for resource-intensive, large-scale simulations: optimization procedures may exhaust resources before interesting, or even feasible, results are obtained; inevitable variations in design specifications and optimization criteria cannot be efficiently accomodated; only limited prior information can be incorporated. Lastly, direct insertion requires a degree of automation that many large-scale simulations do not readily admit.

In the *simulation surrogate* approach, the large-scale numerical simulation is evoked only to construct and validate a simplified model which relates selected design-variable inputs to system-performance outputs; this simplified input–output model then serves as a simulation *surrogate* in ensuing optimization studies. Surrogate techniques enjoy several advantages: optimization procedures will not terminate prematurely; midprocess modifications to design objectives can be efficiently accomodated; simulation results can be effectively recycled; prior information can be gainfully exploited; design interactivity is enhanced; and simulation

*Received by the editors November 30, 1993; accepted for publication (in revised form) March 15, 1995. This work was supported by the Advanced Research Projects Agency under grant N00014-91-J-1889, by the Office of Naval Research under grant N00014-90-J-4124, by the Air Force Office of Scientific Research under grant F49620-94-1-0121, and by NASA under grant NAG 1-1613.

†Department of Nuclear Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139.

‡Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139 (patera@mit.edu).

automation, though desirable, is not necessary. However, the surrogate approach also suffers from several significant disadvantages: simplified input–output models introduce additional prediction errors; the input space definition and dimension must be prescribed "a priori"; design localization deteriorates as the dimension of the input space increases.

Clearly, the direct insertion and surrogate approaches must be treated as complementary and potentially collateral, with inexpensive simulations or final analyses preferring the former, and expensive simulations or exploratory studies selecting the latter. Although continuing advances in parallel computing (Gustafson, Montry, and Benner (1988); Fox et al. (1988); Fischer and Patera (1994)) will certainly render *current* calculations less expensive, increased computational capability will also enable *new* simulations that offer increased physical and engineering relevance. As formerly expensive calculations migrate to the direct insertion class of simulations, formerly "impossible" problems will replenish the surrogate category. There is, therefore, a continuing need for both direct insertion and surrogate approaches; our concern here is with the latter.

The work to date on surrogate procedures for noise-free simulations ranges from early efforts to characterize *average* properties of simulation outputs (McKay, Beckman, and Conover (1979)), to more recent activities focussed on the replication of *detailed* input–output relationships for subsequent application in optimization studies (Sacks, Welch, Mitchell, and Wynn (1989); Sacks, Schiller, and Welch (1989); Cox and John (1992); Morris, Mitchell, and Ylvisaker (1993); Yeşilyurt and Patera (1995)). These surrogate procedures do not, however, explicitly treat the situation in which the simulation output is subject to significant "measurement" noise, and the methods cannot, therefore, address the important issue of bias-variance balance.

Simulation measurement noise—broadly defined as the difference between the (assumed) *deterministic* output of a "continuous" mathematical system and the corresponding (perhaps random) output of an associated approximate solution procedure—arises in many different computational techniques: in finite-difference, finite-element, and finite-volume procedures for partial differential equations, measurement error comprises discretization error, incomplete-iteration error, and arithmetic rounding; in Monte-Carlo techniques for the evaluation of deterministic integrals or the statistics of chaotic or stochastic processes, measurement error takes the more familiar form of estimator variance. In this paper, we propose a nonparametric statistical procedure for the validation and optimization-purposive application of (static) noisy-simulation surrogates; the Bayesian-validated surrogate procedure for noise-free simulations developed in Yeşilyurt and Patera (1995) is, in fact, a special case of the more general framework developed in the current work.

We briefly remark on the relationship between the current work and the much broader field of statistical prediction. The preference for a statistical approach—ostensibly similar to physical-experiment design—to the *noise-free* simulation-surrogate approximation problem reflects, first, the perforce limited samples associated with expensive simulations (this is particularly acute in the absence of computational economies of scale), and second, the lack of regularity information for complex simulation input–output relationships. However, surrogate procedures for noise-free computer experiments differ significantly from statistical prediction techniques for physical experiments in that, for the former, noise mitigation is *not* a factor in efficient "design" (Sacks et al. (1989), Yeşilyurt and Patera (1995)). It may appear that in developing surrogates for noisy simulations we might now appeal more directly to corresponding statistical prediction techniques for physical experiments (Box, Hunter, and Hunter (1978); Seber and Wild (1989); Härdle (1990)).

We believe this is not (or need not be) the case: noisy simulations and noisy physical experiments remain essentially different. In particular, in a typical numerical simulation, first, we need not be concerned with unknown or confounding variables since, by construction, all independent variables are explicit in the formulation of the originating simulation, and second, we can directly control the magnitude of the measurement noise through, for example, mesh refinement or precise replication. Because numerical simulations correspond to a "naive" physical experiment, surrogate procedures for noisy numerical simulations permit both simpler formulation and more explicit error estimation than corresponding methods for physical experiments. For example, nonlinear nonparametric kernel-smoothing regression techniques for physical experiments (Wahba (1975), Silverman (1985), Härdle (1990)) balance predictor bias and measurement variance by an asymptotic relationship between validation sample size and kernel scale; in contrast, the simulation surrogate procedures we propose here balance surrogate bias and measurement error by direct requirements on validation sample size and the (presumed controllable) noise level.

The paper is organized as follows. In §2 we review the optimization-purposiveness structure introduced in Yeşilyurt and Patera (1995), motivate the desired form for the surrogate-validation statement, and introduce our random simulation hypotheses. In §3 we present the validation result and derive the validation error estimate. In §4 we examine the noise contribution to the model prediction error estimator. Lastly, in §5 we present several examples, including a random-media effective-conductivity nested Monte-Carlo finite-element calculation which constitutes a "real," as opposed to contrived, application. We note that in the current paper we take the proposed surrogates as given, "graybox" models; regression procedures by which to generate either blackbox models or corrections to graybox models are discussed in Yeşilyurt and Patera (1995), Yeşilyurt (1995), Otto et al. (1995), and Paraschivoiu (in progress).

2. Optimization purposiveness. We take as given a design-variable M-vector $\mathbf{p}$ in an admissible (closed) design space $\Omega \subset \mathbb{R}^M$, and a (here single) output s which is described by the input–output function $\mathcal{S}(\mathbf{p}) : \Omega \to \mathbb{R}$; we require only that $\mathcal{S}(\mathbf{p}) \in L^\infty(\Omega)$. We assume that there is no *systematic* simulation error: $\mathcal{S}(\mathbf{p})$ is the deterministic input–output function for both the underlying "continuous" mathematical system (e.g., an integral) *and* the numerical simulation in the limit of no measurement noise (e.g., the Monte-Carlo sample mean as the sample size tends to infinity). We next presume that the target value of the output s is prescribed as λ, and that the "best" design point $\mathbf{p}^*$ is defined by

$$(1) \qquad \mathbf{p}^* = \arg \min_{\mathbf{p}\in\Omega} \Phi(\mathbf{p}, \lambda) \,,$$

where

$$(2) \qquad \Phi(\mathbf{p}, \lambda) = |\mathcal{S}(\mathbf{p}) - \lambda| \,.$$

A much more general optimization framework, in which the objective function, $\Phi(\mathbf{p}, \lambda)$, is an arbitrary function of the simulation input–output relationship $\mathcal{S}(\mathbf{p})$, is developed in Yeşilyurt and Patera (1995); the specific case given here, related to the problem of "discrimination" (Seber and Wild (1989)), leads to a simpler exposition which is adequate for our current purposes.

We next introduce the simulation surrogate, $\widetilde{\mathcal{S}}(\mathbf{p})$, which should, first, admit considerably simpler evaluation than the original input–output function, $\mathcal{S}(\mathbf{p})$, and, second, provide a reasonable approximation to $\mathcal{S}(\mathbf{p})$ over the design space Ω. We then replace $\mathcal{S}(\mathbf{p})$ in (2) with $\widetilde{\mathcal{S}}(\mathbf{p})$ to construct the approximate design problem

$$(3) \qquad \widetilde{\mathbf{p}}^* = \arg \min_{\mathbf{p}\in\Omega} |\widetilde{\mathcal{S}}(\mathbf{p}) - \lambda| \,;$$

the discussion that follows is perhaps clearest when $\widetilde{\mathcal{S}}(\widetilde{\mathbf{p}}^*) = \lambda$. The advantage of (3) over (1) is, of course, more ready and certain solution; perhaps more importantly, midprocess modifications in design specifications, here represented by λ, can be treated without re-appeal to $\mathcal{S}(\mathbf{p})$. We must, however, understand how errors in the surrogate will be reflected in predicted system configuration and performance.

We describe a particular surrogate-validation error estimate which leads to practically relevant purposive statements. We first assume, given small positive real parameters $0 < \varepsilon_1 < 1$ and $0 < \varepsilon_2 < 1$, and an importance function $\rho(\mathbf{p})$ which is strictly positive and integrates to unity over Ω, that we can find a model prediction error estimator B such that

$$\Pr\left\{\int_{\{\mathbf{p}\in\Omega\,|\,|\mathcal{S}(\mathbf{p})-\widetilde{\mathcal{S}}(\mathbf{p})|\leq B\}} \rho(\mathbf{p})\mathbf{dp} \geq 1-\varepsilon_1\right\} \geq 1-\varepsilon_2\,, \tag{4}$$

where Pr{*event*} refers to the probability of *event*, and **dp** is a differential element of Ω. The result (4) can be viewed as a probably approximately correct statement (Valiant (1984), Gallant (1990)), and is also related to the probabilistic framework of information-based complexity theory (Traub, Wasilkowski, and Wozniakowski (1988)). In words, this validation error statement indicates that, over some volumetrically important subdomain of Ω, the surrogate L^∞-prediction error is less than B with high probability.

It can then readily be shown (Yeşilyurt and Patera (1995), Yeşilyurt (1995)) that, with probability greater than $1-\varepsilon_2$, for any region $\mathcal{R} \subset \Omega$ for which $\int_{\mathcal{R}} \rho(\mathbf{p})\mathbf{dp} > \varepsilon_1$, there exists a (in fact, many) $\hat{\mathbf{p}}^* \in \mathcal{R}$ for which

$$|\mathcal{S}(\hat{\mathbf{p}}^*) - \lambda| \leq |\widetilde{\mathcal{S}}(\widetilde{\mathbf{p}}^*) - \lambda| + B + \delta_{\mathcal{R}}\,, \tag{5}$$

where

$$\delta_{\mathcal{R}} = \max_{\mathbf{p}\in\mathcal{R}} |\,\widetilde{\mathcal{S}}(\mathbf{p}) - \widetilde{\mathcal{S}}(\widetilde{\mathbf{p}}^*)\,|\,. \tag{6}$$

The choice of $\mathcal{R}$ may be optimized with respect to various metrics (Otto et al. (1995)); for example, in Yeşilyurt and Patera (1995), $\mathcal{R}$ is chosen as the region $\mathcal{R}'$ of "size" $\nu\varepsilon_1$ ($1 < \nu \leq 1/\varepsilon_1$ given) that minimizes $\delta_{\mathcal{R}'}$. From (5) we deduce, assuming minimal continuity requirements on $\mathcal{S}(\mathbf{p})$ and $\widetilde{\mathcal{S}}(\mathbf{p})$, that for sufficiently accurate $\widetilde{\mathcal{S}}(\mathbf{p})$ (small B) and sufficiently small ε_1 (small $\delta_{\mathcal{R}}$), there exist many design points $\hat{\mathbf{p}}^*$ arbitrarily near $\widetilde{\mathbf{p}}^*$ at which *actual* system performance, as measured by $|\mathcal{S}(\hat{\mathbf{p}}^*) - \lambda|$, is arbitrarily close to the optimal system performance predicted by the surrogate. If we further presume that $|\mathcal{S}(\mathbf{p}) - \lambda|$ and $|\widetilde{\mathcal{S}}(\mathbf{p}) - \lambda|$ are quasi-convex, we can construct, based solely on the surrogate $\widetilde{\mathcal{S}}(\mathbf{p})$, a random region $\mathcal{K}$ that shrinks to $\widetilde{\mathbf{p}}^*$ as $B \to 0$, $\varepsilon_1 \to 0$, and that, with probability greater than $1-\varepsilon_2$, contains the *actual* design point, $\mathbf{p}^*$ (Yeşilyurt and Patera (1995), Yeşilyurt (1995)).

The many practical ramifications of these purposiveness estimates are described in §5 of this paper and, more extensively, in Yeşilyurt and Patera (1995) and Otto et al. (1995); other approaches to optimization purposiveness in the context of nonlinear nonparametric regression can be found in Silverman (1985) and Härdle (1990, Chap. 8). Note that, although our estimates in volume are independent of input dimension M, the variations in individual components of $\mathbf{p}$ will increase with increasing M. As a consequence, our methods are viable for large M only if the inputs are strongly correlated through $\rho(\mathbf{p})$, as in shape optimization (Otto et al. (1995), Otto (in progress)), or if screening techniques are applied prior to surrogate construction-validation (Welch et al. (1992)).

It remains to find a validation algorithm which yields the error estimate (4). For the case in which there is no measurement error in the simulation (that is, given a $\mathbf{p} \in \Omega$, $s = \mathcal{S}(\mathbf{p})$ can be determined with infinite precision), the algorithm and associated proof are very simple

(Yeşilyurt and Patera (1995)). We consider here the slightly more complicated case in which the simulation result is subject to measurement noise. More precisely, we shall assume that our simulation output is now a random variable, R, characterized by mean $\mathcal{S}(\mathbf{p})$, the deterministic input–output function of the underlying exact (unapproximated) mathematical model; finite variance $\sigma_W(\mathbf{p})$; and scale-parametrized (smooth) conditional probability density

$$f_{R|\mathbf{P}}(r|\mathbf{p}) = \frac{1}{\sigma_W(\mathbf{p})} g\left(\frac{r - \mathcal{S}(\mathbf{p})}{\sigma_W(\mathbf{p})}\right) . \tag{7}$$

Equivalently, we can express the simulation output as

$$R = \mathcal{S}(\mathbf{p}) + W , \tag{8}$$

where W is a (possibly unbounded) absolutely continuous measurement noise random variable with probability density

$$f_{W|\mathbf{P}}(w|\mathbf{p}) = \frac{1}{\sigma_W(\mathbf{p})} g\left(\frac{w}{\sigma_W(\mathbf{p})}\right) . \tag{9}$$

As the expectation of R given $\mathbf{p}$ is $\mathcal{S}(\mathbf{p})$, or, equivalently, the expectation of W is zero, we implicitly preclude biased simulation estimators; we shall further assume that the noise is symmetric, $g(w) = g(-w)$. These assumptions, though not essential, greatly facilitate the subsequent analysis, in particular in the case of unbounded noise. Note that we aim to predict the exact mathematical-model output, $\mathcal{S}(\mathbf{p})$, not the simulation result, R, since the simulation measurement error is presumed not relevant to the mathematical system which the numerical calculation purportedly represents.

3. Validation procedure. In this section we present (in §3.1) and prove (in §3.2) our central validation result for noisy simulations.

3.1. Validation statement. We take as given $0 < \varepsilon_1 < 1, 0 < \varepsilon_2 < 1$, and $0 \leq \alpha \leq 1$, and define $\gamma_\alpha(\mathbf{p})$ and N as

$$\int_{-\infty}^{-\gamma_\alpha(\mathbf{p})} f_{W|\mathbf{P}}(w|\mathbf{p}) dw = \alpha \tag{10}$$

and

$$N = \frac{\ln \varepsilon_2}{\ln[1 - \varepsilon_1(1 - \alpha)]} , \tag{11}$$

respectively. We next construct the sample of independent identically distributed ordered pairs

$$\{(\mathbf{P}_1, R_1), \ldots, (\mathbf{P}_N, R_N)\} , \tag{12}$$

where each $(\mathbf{P}_j, R_j)$ has joint probability density function

$$f_{\mathbf{P},R}(\mathbf{p}, r) = \rho(\mathbf{p}) f_{R|\mathbf{P}}(r|\mathbf{p}) ; \tag{13}$$

the importance function-cum-marginal density $\rho(\mathbf{p})$ can thus be viewed as a Bayesian prior on the simulation input vector (Yeşilyurt and Patera (1995)). Then,

$$\Pr\left\{\int_{\{\mathbf{p}\in\Omega \,|\, |\mathcal{S}(\mathbf{p}) - \widetilde{\mathcal{S}}(\mathbf{p})| \leq U\}} \rho(\mathbf{p}) \mathbf{dp} \geq 1 - \varepsilon_1\right\} \geq 1 - \varepsilon_2 , \tag{14}$$

where

$$U = \max_{j\in\{1,\dots,N\}} \{|R_j - \widetilde{\mathcal{S}}(\mathbf{P}_j)| + \gamma_\alpha(\mathbf{P}_j)\} \tag{15}$$

is the noisy *model prediction error estimator* (that is, the surrogate "bias"). Identifying $U = B$, we recognize the probably approximately correct error statement, (4), motivated in §2.

The Monte-Carlo algorithm associated with (14) and (15) is transparent: we sample N values of the design random vector over Ω, $\mathbf{P}_j$, $j = 1, \dots, N$, according to the specified density $\rho(\mathbf{p})$; we compute a (noisy) simulation result, R_j, at each design point $\mathbf{P}_j$; finally, we calculate the model prediction error estimator, U, from (10) and (15). The result can be interpreted from two perspectives: as a tolerance limit, U, for $|\mathcal{S}(\mathbf{P}) - \widetilde{\mathcal{S}}(\mathbf{P})|$; and as a nonparametric joint confidence interval over $1-\varepsilon_1$ of the design domain, Ω, for the surrogate-simulation misfit, $-U \le \mathcal{S}(\mathbf{p}) - \widetilde{\mathcal{S}}(\mathbf{p}) \le U$, or, equivalently, for the noise-free simulation result, $\widetilde{\mathcal{S}}(\mathbf{p}) - U \le \mathcal{S}(\mathbf{p}) \le \widetilde{\mathcal{S}}(\mathbf{p}) + U$. We now turn to a proof of (14).

3.2. Proof of validation error estimate. To begin, we define ξ_q as the q "quantile" of $|\mathcal{S}(\mathbf{p}) - \widetilde{\mathcal{S}}(\mathbf{p})|$,

$$\xi_q = \min \zeta \text{ such that } \mathcal{F}(\zeta) \ge q \ , \tag{16}$$

where

$$\mathcal{F}(\zeta) = \int_{\{\mathbf{p}\in\Omega \,|\, |\mathcal{S}(\mathbf{p})-\widetilde{\mathcal{S}}(\mathbf{p})|\le\zeta\}} \rho(\mathbf{p})\mathbf{dp} \ . \tag{17}$$

We then introduce the domain

$$\mathcal{Q} = \{\mathbf{p} \in \Omega \,|\, |\mathcal{S}(\mathbf{p}) - \widetilde{\mathcal{S}}(\mathbf{p})| \ge \xi_{1-\varepsilon_1}\} \tag{18}$$

and associated partition

$$\mathcal{Q} = \mathcal{Q}^+ \cup \mathcal{Q}^- \ , \tag{19}$$

where

$$\begin{aligned} \mathcal{Q}^+ &= \{\mathbf{p} \in \mathcal{Q} \,|\, \mathcal{S}(\mathbf{p}) - \widetilde{\mathcal{S}}(\mathbf{p}) \ge 0\} \ , \\ \mathcal{Q}^- &= \{\mathbf{p} \in \mathcal{Q} \,|\, \mathcal{S}(\mathbf{p}) - \widetilde{\mathcal{S}}(\mathbf{p}) < 0\} \ . \end{aligned} \tag{20}$$

It then follows from (16), (18), and the right-continuity of $\mathcal{F}(\zeta)$, that

$$\int_{\mathcal{Q}} \rho(\mathbf{p})\mathbf{dp} = \beta\varepsilon_1 \tag{21}$$

for some β, $1 \le \beta \le 1/\varepsilon_1$. Note that, if $\mathcal{F}(\zeta)$ is continuous (that is, no contours of $|\mathcal{S}(\mathbf{p}) - \widetilde{\mathcal{S}}(\mathbf{p})|$ are of finite measure), then β will be unity.

We now define $(\mathbf{p}, w) \in \mathcal{D} \equiv \Omega \times \mathbb{R}$, and introduce $\mathcal{T} \subset \mathcal{D}$,

$$\begin{aligned} \mathcal{T}^+ &= \{(\mathbf{p}, w) \in \mathcal{D} \,|\, \mathbf{p} \in \mathcal{Q}^+, w \ge -\gamma_\alpha(\mathbf{p})\}, \\ \mathcal{T}^- &= \{(\mathbf{p}, w) \in \mathcal{D} \,|\, \mathbf{p} \in \mathcal{Q}^-, w \le \gamma_\alpha(\mathbf{p})\}, \\ \mathcal{T} &= \mathcal{T}^+ \cup \mathcal{T}^-. \end{aligned} \tag{22}$$

It then follows from (21) and (22) that

$$\begin{aligned}
&\int_{\mathcal{T}} \rho(\mathbf{p}) f_{W|\mathbf{P}}(w|\mathbf{p}) dw\, \mathbf{dp} \\
&= \int_{\mathcal{Q}^+} \rho(\mathbf{p}) \left(\int_{-\gamma_\alpha(\mathbf{p})}^{\infty} f_{W|\mathbf{P}}(w|\mathbf{p}) dw \right) \mathbf{dp} + \int_{\mathcal{Q}^-} \rho(\mathbf{p}) \left(\int_{-\infty}^{\gamma_\alpha(\mathbf{p})} f_{W|\mathbf{P}}(w|\mathbf{p}) dw \right) \mathbf{dp} \\
&= \int_{\mathcal{Q}} \rho(\mathbf{p}) \mathbf{dp}\, (1-\alpha) = \beta\varepsilon_1(1-\alpha) \ . \qquad (23)
\end{aligned}$$

We now introduce $W_j = R_j - \mathcal{S}(\mathbf{P}_j)$, the noise contribution associated with the $\mathbf{P}_j$ sample point; from (7)–(9) and (13) we know that $(\mathbf{P}_j, W_j)$ has joint probability density function

$$f_{\mathbf{P},W}(\mathbf{p}, w) = \rho(\mathbf{p}) f_{W|\mathbf{P}}(w|\mathbf{p}) \ . \qquad (24)$$

We therefore infer from (23) and (24) that

$$\begin{aligned}
&\Pr\{\exists\, j^* \in \{1, \dots, N\} \text{ such that } (\mathbf{P}_{j^*}, W_{j^*}) \in \mathcal{T}\} \\
&= 1 - (1 - \beta\varepsilon_1(1-\alpha))^N \geq 1 - (1 - \varepsilon_1(1-\alpha))^N \ , \qquad (25)
\end{aligned}$$

and thus

$$\Pr\{\exists\, j^* \in \{1, \dots, N\} \text{ such that } (\mathbf{P}_{j^*}, W_{j^*}) \in \mathcal{T}\} \geq 1 - \varepsilon_2 \qquad (26)$$

for N satisfying the sample-size requirement (11).

To complete the demonstration we note from (18) and (22) that, for $(\mathbf{P}_{j^*}, W_{j^*}) \in \mathcal{T}$,

$$\begin{aligned}
|R_{j^*} - \widetilde{\mathcal{S}}(\mathbf{P}_{j^*})| + \gamma_\alpha(\mathbf{P}_{j^*}) &= |\mathcal{S}(\mathbf{P}_{j^*}) - \widetilde{\mathcal{S}}(\mathbf{P}_{j^*}) + W_{j^*}| + \gamma_\alpha(\mathbf{P}_{j^*}) \\
&\geq |\mathcal{S}(\mathbf{P}_{j^*}) - \widetilde{\mathcal{S}}(\mathbf{P}_{j^*})| \geq \xi_{1-\varepsilon_1} \ . \qquad (27)
\end{aligned}$$

Therefore, from (16) and (27)

$$\begin{aligned}
&\int_{\{\mathbf{p}\in\Omega \,|\, |\mathcal{S}(\mathbf{p}) - \widetilde{\mathcal{S}}(\mathbf{p})| \leq |R_{j^*} - \widetilde{\mathcal{S}}(\mathbf{P}_{j^*})| + \gamma_\alpha(\mathbf{P}_{j^*})\}} \rho(\mathbf{p}) \mathbf{dp} \\
&\geq \int_{\{\mathbf{p}\in\Omega \,|\, |\mathcal{S}(\mathbf{p}) - \widetilde{\mathcal{S}}(\mathbf{p})| \leq \xi_{1-\varepsilon_1}\}} \rho(\mathbf{p}) \mathbf{dp} \geq 1 - \varepsilon_1 \ . \qquad (28)
\end{aligned}$$

Finally, (14) follows from

$$\begin{aligned}
&\int_{\{\mathbf{p}\in\Omega \,|\, |\mathcal{S}(\mathbf{p}) - \widetilde{\mathcal{S}}(\mathbf{p})| \leq |R_{j^*} - \widetilde{\mathcal{S}}(\mathbf{P}_{j^*})| + \gamma_\alpha(\mathbf{P}_{j^*})\}} \rho(\mathbf{p}) \mathbf{dp} \geq 1 - \varepsilon_1 \\
&\Rightarrow \int_{\{\mathbf{p}\in\Omega \,|\, |\mathcal{S}(\mathbf{p}) - \widetilde{\mathcal{S}}(\mathbf{p})| \leq U\}} \rho(\mathbf{p}) \mathbf{dp} \geq 1 - \varepsilon_1 \ , \qquad (29)
\end{aligned}$$

and the probability statement (26). This result can be extended to multiple outputs by application of Boole's inequality (Yeşilyurt (1995)). In summary, the proof is simply a binomial argument in a product space comprising both the input and the noise contributions to the simulation result.

4. Model prediction error estimator. In this section we aim to understand the effect of the simulation noise on the model prediction error estimator, U. In particular, improved validation (smaller ε_1 and ε_2) implies, from (11), larger N, and hence greater likelihood that U, the $|R_j - \tilde{\mathcal{S}}(\mathbf{P}_j)|$ sample *maximum*, will reflect—in fact, select—larger noise contributions. We remark that the rigorous application of the validation statement (14) to the purposive structure described in §2 is not dependent on estimation of the noise; our noise estimates serve only to minimize the computational effort or to sharpen the bounds. In this sense, we can proceed somewhat less stringently, looking at expectation rather than confidence intervals.

4.1. Bounded noise. We consider here the case in which the noise is bounded: $f_{W|\mathbf{P}}(w|\mathbf{p}) = 0$ for $|w| > (c_W/2)\sigma_W(\mathbf{p})$, where $c_W > 0$ is independent of $\mathbf{p}$; equivalently, $g(v) = 0$ for $|v| > c_W/2$. Bounded noise is a reasonable model (or bound) for the measurement noise associated with finite-element methods for partial differential equations. For bounded noise it is interesting to consider $\alpha = 0$, which implies from (10) and (11) that

$$\gamma_\alpha(\mathbf{p}) = (c_W/2)\sigma_W(\mathbf{p}) \tag{30}$$

and

$$N = \frac{\ln \varepsilon_2}{\ln(1-\varepsilon_1)} , \tag{31}$$

respectively. Finally, from (8) and (15) we derive that

$$U \le \max_{j\in\{1,\dots,N\}} |\mathcal{S}(\mathbf{P}_j) - \tilde{\mathcal{S}}(\mathbf{P}_j)| + c_W \sigma_W^{\max} , \tag{32}$$

where $\sigma_W^{\max} = \max_{\mathbf{p}\in\Omega} \sigma_W(\mathbf{p})$. In the limit that $\sigma_W^{\max} \to 0$, both the sample size requirement, (31), and the model prediction error estimator, (32), approach the corresponding limits for a *noise-free* simulation (Yeşilyurt and Patera (1995)); at least in this particular limit, our noisy result is, in some sense, sharp. Biased bounded noise is considered in Yeşilyurt (1995).

4.2. Unbounded noise. We consider here the case of unbounded noise, in which the probability density for the noise is potentially nonzero for all values of $w \in \mathbb{R}$. To make our ideas more concrete, we shall assume that $\alpha = 1/2$, so that $\gamma_\alpha(\mathbf{p}) = 0$ (independent of both $\mathbf{p}$ and the form of the assumed-symmetric $g(w)$), and

$$N = \frac{\ln \varepsilon_2}{\ln(1-\frac{\varepsilon_1}{2})} . \tag{33}$$

Unbounded noise is relevant to Monte-Carlo simulations (Rubinstein (1981)). We will occasionally appeal to a canonical "Monte-Carlo subproblem," by which we shall refer to a specific simulation procedure in which the simulation random output is the result of a standard sample mean calculation,

$$R = \frac{1}{\mathcal{N}} \sum_{k=1}^{\mathcal{N}} Q_k , \tag{34}$$

where the Q_k are (for any particular $\mathbf{p} \in \Omega$) independent identically distributed random variables with mean $\mathcal{S}(\mathbf{p})$, finite variance $\sigma_Q^2(\mathbf{p})$, and probability density function $f_Q(q)$. It follows that $\sigma_W(\mathbf{p}) = \sigma_Q(\mathbf{p})/\sqrt{\mathcal{N}}$, and that R is asymptotically normal as $\mathcal{N} \to \infty$. We refer to $\mathcal{N}$ as the subproblem sample size to avoid confusion with the validation sample size, N.

We measure the effect of the noise by the random variable $Y = \mathcal{Y}(\underline{\mathbf{P}}, \underline{W})$,

$$\mathcal{Y}(\underline{\mathbf{P}}, \underline{W}) = \max_{j\in\{1,\ldots,N\}} |\mathcal{S}(\mathbf{P}_j) + W_j - \widetilde{\mathcal{S}}(\mathbf{P}_j)| - \max_{j'\in\{1,\ldots,N\}} |\mathcal{S}(\mathbf{P}_{j'}) - \widetilde{\mathcal{S}}(\mathbf{P}_{j'})| \,, \tag{35}$$

where $\underline{\mathbf{P}} = (\mathbf{P}_1, \ldots, \mathbf{P}_N)$ and $\underline{W} = (W_1, \ldots, W_N)$. Note that, even in the absence of noise, our validation statement does not permit a priori estimation of $|\mathcal{S}(\mathbf{p}) - \widetilde{\mathcal{S}}(\mathbf{p})|$ without further assumptions on $\mathcal{S}(\mathbf{p})$ and $\widetilde{\mathcal{S}}(\mathbf{p})$; our focus here is on understanding the incremental effect of noise on the model prediction error estimator. (The presence of unbounded noise also increases N, (33), with respect to the bounded-noise and noise-free cases, (31); this "secondary" effect is not reflected in Y.) Ultimately, we will require a relationship which specifies the allowable simulation noise, as a function of ε_1 and ε_2, such that "variance" does not dominate surrogate "bias." We implicitly assume that the simulation noise can be controlled through precise random sampling and subproblem replication, as in our model Monte-Carlo subproblem (34); if this were not the case, we would need to pursue a kernel smoothing approach (Wahba (1975), Silverman (1985), Härdle (1990)), in which local-in-$\mathbf{p}$ validation averages (or, equivalently, smoothness requirements) replace subproblem replication.

4.2.1. Bound for expected noise contribution. We consider here estimates for the expectation of Y with respect to $\underline{\mathbf{P}}$ and $\underline{W}$ *conditioned* on the event that $A = 1$, where A is the indicator random variable,

$$A = \begin{cases} 0 & \text{if} \quad \int_{\{\mathbf{p}\in\Omega\,|\,|\mathcal{S}(\mathbf{p})-\widetilde{\mathcal{S}}(\mathbf{p})|\le U\}} \rho(\mathbf{p})\mathbf{dp} < 1-\varepsilon_1 \\ 1 & \text{if} \quad \int_{\{\mathbf{p}\in\Omega\,|\,|\mathcal{S}(\mathbf{p})-\widetilde{\mathcal{S}}(\mathbf{p})|\le U\}} \rho(\mathbf{p})\mathbf{dp} \ge 1-\varepsilon_1, \end{cases} \tag{36}$$

that signals successful validation. Denoting the sample space as $\Xi = \Omega^N \times \mathbb{R}^N$, we can express the required conditional expectation as

$$E_{\underline{\mathbf{P}},\underline{W}|A=1}(Y) = \frac{\int_{\{(\underline{\mathbf{p}},\underline{w})\in\,\Xi\,|\,A=1\}} \mathcal{Y}(\underline{\mathbf{p}},\underline{w}) \prod_{i=1}^N f_{\mathbf{P},W}(\mathbf{p}_i, w_i)\mathbf{dp}_i dw_i}{\int_{\{(\underline{\mathbf{p}},\underline{w})\in\,\Xi\,|\,A=1\}} \prod_{i=1}^N f_{\mathbf{P},W}(\mathbf{p}_i, w_i)\mathbf{dp}_i dw_i} \,. \tag{37}$$

Denoting

$$\hat{j} = \arg \max_{j\in\{1,\ldots,N\}} |\mathcal{S}(\mathbf{P}_j) + W_j - \widetilde{\mathcal{S}}(\mathbf{P}_j)|, \tag{38}$$

we deduce that

$$\begin{aligned} Y &\le |\mathcal{S}(\mathbf{P}_{\hat{j}}) + W_{\hat{j}} - \widetilde{\mathcal{S}}(\mathbf{P}_{\hat{j}})| - |\mathcal{S}(\mathbf{P}_{\hat{j}}) - \widetilde{\mathcal{S}}(\mathbf{P}_{\hat{j}})| \\ &\le |W_{\hat{j}}| \le Z \,, \end{aligned} \tag{39}$$

where $Z = \mathcal{Z}(\underline{\mathbf{P}}, \underline{W})$ is the random variable

$$Z = \max_{j\in\{1,\ldots,N\}} (|W_j|) \,. \tag{40}$$

Since Z is nonnegative, and since the denominator of (37) is, by (14), greater than or equal to $1 - \varepsilon_2$, we conclude that

$$\begin{aligned} E_{\underline{\mathbf{P}},\underline{W}|A=1}(Y) &\le \int_{\{(\underline{\mathbf{p}},\underline{w})\in\,\Xi\,|\,A=1\}} \mathcal{Z}(\underline{\mathbf{p}},\underline{w}) \prod_{i=1}^N f_{\mathbf{P},W}(\mathbf{p}_i, w_i)\mathbf{dp}_i dw_i/(1-\varepsilon_2) \\ &\le \int_{\{(\underline{\mathbf{p}},\underline{w})\in\,\Xi\}} \mathcal{Z}(\underline{\mathbf{p}},\underline{w}) \prod_{i=1}^N f_{\mathbf{P},W}(\mathbf{p}_i, w_i)\mathbf{dp}_i dw_i/(1-\varepsilon_2) \\ &= E_{\underline{\mathbf{P}},\underline{W}}(Z)/(1-\varepsilon_2) \,. \end{aligned} \tag{41}$$

Finally, applying standard conditional-expectation and order-statistic manipulations (Mood, Graybill, and Boes (1974); David (1981); Arnold and Balakrishnan (1989)), we arrive at the explicit expression

$$E_{\underline{\mathbf{P}},\underline{W}}(Z) = \int_{\Omega^N} \prod_{k=1}^{N} \rho(\mathbf{p}_k) \times \left(\int_0^\infty \sum_{i=1}^{N} \prod_{j \neq i} (1 - 2F_{W|\mathbf{P}}(-v|\mathbf{p}_j)) 2 f_{W|\mathbf{P}}(v|\mathbf{p}_i) v \, dv \right) \mathbf{dp}_k \,, \tag{42}$$

where $F_{W|\mathbf{P}}(w|\mathbf{p})$ is the cumulative distribution function associated with $f_{W|\mathbf{P}}(w|\mathbf{p})$. Expression (42) now admits a priori treatment since all dependence on the uncharacterized functions $\mathcal{S}(\mathbf{p})$ and $\widetilde{\mathcal{S}}(\mathbf{p})$ has been eliminated.

Estimation for unknown density. We consider here a further bound which requires knowledge only of the (assumed-finite) variance, $\sigma_W^2(\mathbf{p})$, of the noise conditional density $f_{W|\mathbf{P}}$. Applying the Cauchy–Schwarz inequality to the integral over v in (42), we find

$$E_{\underline{\mathbf{P}},\underline{W}}(Z) \leq \int_{\Omega^N} \prod_{k=1}^{N} \rho(\mathbf{p}_k) \times \sum_{i=1}^{N} \left(\int_0^\infty \prod_{j \neq i} (1 - 2F_{W|\mathbf{P}}(-v|\mathbf{p}_j))^2 \, 2 f_{W|\mathbf{P}}(v|\mathbf{p}_i) dv \right)^{1/2} \sigma_W(\mathbf{p}_i) \mathbf{dp}_k . \tag{43}$$

Next we apply the Cauchy–Schwarz inequality to the sum over i to obtain

$$E_{\underline{\mathbf{P}},\underline{W}}(Z) \leq \int_{\Omega^N} \prod_{k=1}^{N} \rho(\mathbf{p}_k) \times \left(\int_0^\infty \sum_{i=1}^{N} \prod_{j \neq i} (1 - 2F_{W|\mathbf{P}}(-v|\mathbf{p}_j))^2 \, 2 f_{W|\mathbf{P}}(v|\mathbf{p}_i) dv \right)^{1/2} \times \left(\sum_{\ell=1}^{N} \sigma_W^2(\mathbf{p}_\ell) \right)^{1/2} \mathbf{dp}_k . \tag{44}$$

Noting that we can conservatively (albeit crudely) replace the exponent of the $(1 - 2F_{W|\mathbf{P}}(-v|\mathbf{p}_j))$ term in the integral over v by unity (this is not required if $\sigma_W(\mathbf{p})$ is constant), we find

$$E_{\underline{\mathbf{P}},\underline{W}}(Z) \leq \int_{\Omega^N} \prod_{k=1}^{N} \rho(\mathbf{p}_k) \left(\sum_{\ell=1}^{N} \sigma_W^2(\mathbf{p}_\ell) \right)^{1/2} \mathbf{dp}_k \,. \tag{45}$$

A final application of the Cauchy–Schwarz inequality then yields

$$E_{\underline{\mathbf{P}},\underline{W}}(Z) \leq \sqrt{N} \sigma_W^\rho \tag{46}$$

where

$$\sigma_W^\rho = \left(\int_\Omega \rho(\mathbf{p}) \sigma_W^2(\mathbf{p}) \mathbf{dp} \right)^{1/2} . \tag{47}$$

In cases where $\sigma_W(\mathbf{p})$ is known only at the validation sample points, the integral (47) is approximated by an N-point Monte-Carlo quadrature,

$$\sigma_W^\rho \approx \left(\frac{1}{N} \sum_{j=1}^{N} \sigma_W^2(\mathbf{p}_j) \right)^{1/2} , \tag{48}$$

where $\{\mathbf{p}_1, \ldots, \mathbf{p}_N\}$ is the validation input sample realization. In the case where $\sigma_W(\mathbf{p})$ is, in fact, constant, the prefactor in (46) is easily improved to

$$\sqrt{N/2}\sqrt{2N/(2N-1)}.$$

In effect, (46), (47) extends certain standard results for extreme values (David (1981), Arnold and Balakrishnan (1989)) to the case of "marginal," or noisy (Ho, Sreenivas, and Vakili (1992)), order statistics. The bound (46), (47) indicates that, as expected, isolated noisy regions of the input space Ω will not unduly contaminate the model prediction error estimator. This result also suggests heuristics for designing computationally efficient "coupled" subproblem-validation evaluation procedures. For example, if we generalize (34) to permit $\mathbf{p}$-variable subproblem sample size, $\mathcal{N}(\mathbf{p})$, a simple constrained optimization exercise demonstrates that σ_W^ρ is minimized at fixed computational cost—defined as $\sum_{j=1}^N \mathcal{N}(\mathbf{p}_j)$—for $\mathcal{N}(\mathbf{p}_j) \propto \sigma_Q(\mathbf{p}_j)$. This particular coupled strategy is of limited computational relevance, as (46), (47) is not sharp for the noise densities typically encountered in practice.

Our fully assembled bound, (41), (46), and (47) (or (48)), states that σ_W^ρ must decrease no more slowly than $1/\sqrt{N}$ if the noise contribution to the model prediction error is not to dominate. For example, for our Monte-Carlo subproblem, (34), we need to increase the subproblem sample size, $\mathcal{N}$, roughly in proportion to the validation sample size, N, to bound the noise contribution as $N \to \infty$. Rephrasing our condition in terms of ε_1 and ε_2 from (33), we require that

$$\frac{\sigma_W^\rho\sqrt{-\ln\varepsilon_2}}{(1-\varepsilon_2)\sqrt{\varepsilon_1/2}} \tag{49}$$

remain bounded as $\varepsilon_1 \to 0$, $\varepsilon_2 \to 0$ (note we approximate $\ln(1-\varepsilon_1/2)$ by $(-\varepsilon_1/2)$). Although the dependence on ε_1 and certainly ε_2 is relatively weak, (49) is pessimistic for most noise densities that will obtain in practice. □

Estimation for known density. In the event that we know (or can approximate) the noise conditional density, $f_{W|\mathbf{P}}$, we can directly evaluate (42) as

$$E_{\underline{\mathbf{P}},\underline{W}}(Z) = N\int_0^\infty \mathcal{I}_1(v)^{N-1}\mathcal{I}_2(v)v\,dv\ , \tag{50}$$

where

$$\mathcal{I}_1(v) = \int_\Omega \rho(\mathbf{p})(1-2F_{W|\mathbf{P}}(-v|\mathbf{p}))\mathbf{dp}\ , \tag{51}$$

$$\mathcal{I}_2(v) = \int_\Omega \rho(\mathbf{p})2f_{W|\mathbf{P}}(v|\mathbf{p})\mathbf{dp}\ . \tag{52}$$

In cases where $f_{W|\mathbf{P}}$ is known only at the validation sample points, the integrals (51) and (52) are effected by an N-point Monte-Carlo quadrature,

$$\mathcal{I}_1(v) \approx \frac{1}{N}\sum_{j=1}^N (1-2F_{W|\mathbf{P}}(-v|\mathbf{p}_j))\ , \tag{53}$$

$$\mathcal{I}_2(v) \approx \frac{1}{N}\sum_{j=1}^N 2f_{W|\mathbf{P}}(v|\mathbf{p}_j)\ , \tag{54}$$

where $\{\mathbf{p}_1, \ldots, \mathbf{p}_N\}$ is the validation input sample realization. Our fully assembled bound is, thus, (41), (50), and (51)–(52) or (53)–(54), in which (50) is evaluated by numerical or symbolic integration. □

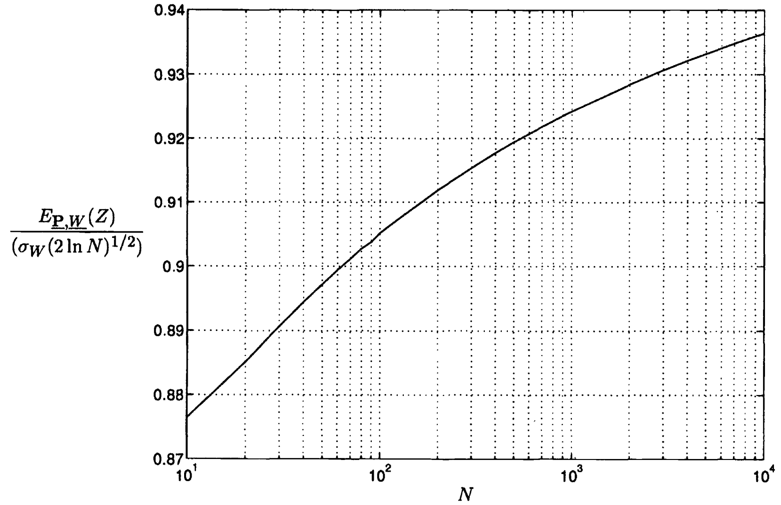


FIG. 1. *Plot of $E_{\underline{\mathbf{P}},\underline{W}}(Z)/(\sigma_W(2\ln N)^{1/2})$ as a function of sample size N for the normal parent.*

Estimation for normal parent. We briefly discuss the application of the bound (41), (50), (51)–(52) to the particular case in which the noise density is normal, first, because of the central importance of the normal density in many Monte-Carlo calculations, and second, to demonstrate that, in practice, (49) is unduly pessimistic. We consider a particularly simple situation, $\sigma_W(\mathbf{p}) = \sigma_W$ constant, for which (50) admits ready numerical evaluation. In Figure 1 we plot $E_{\underline{\mathbf{P}},\underline{W}}(Z)/(\sigma_W(2\ln N)^{1/2})$ as a function of sample size for $10 \leq N \leq 10^4$. As expected from asymptotic theory (David (1981)), this normalized expectation slowly approaches unity as N increases. We thus conclude that, if the noise contribution is not to dominate the model prediction error estimator, σ_W must decrease no more slowly than $(2\ln N)^{-1/2}$. More precisely, from (33), (41), (50), and Figure 1, we require that

$$\text{(55)} \qquad \frac{\sqrt{2}\sigma_W}{1-\varepsilon_2}\left(\ln(-\ln\varepsilon_2) + \ln\frac{2}{\varepsilon_1}\right)^{1/2}$$

remain bounded as $\varepsilon_1 \to 0$, $\varepsilon_2 \to 0$. This condition is clearly much less stringent than (49). (It is certainly of interest to develop an asymptotic theory for the normal parent $E_{\underline{\mathbf{P}},\underline{W}}(Z)$ which includes variance-variability ($\sigma_W(\mathbf{p}) \neq$ constant), as this would permit the design of practically relevant optimally coupled subproblem-validation evaluation procedures; to date, however, we have been unable to obtain a useful expression.) □

We conclude by remarking that our estimates of this section and §3 are similar to—but simpler than—corresponding $L^\infty(\Omega)$ estimates for the kernel smoothing approach (Härdle (1990)). Our results are simpler because we are less ambitious: first, we obtain estimates only for a "holey" (or "quantile") L^∞-seminorm which admits the possibility of an uncharacterized region of relative volume as large as ε_1 (Traub, Wasilkowski, and Wozniakowski (1988)); and second, we rely on precise sampling and replication for noise control. An advantage of the "holey" seminorm is that we require only minimal regularity hypotheses on $\mathcal{S}(\mathbf{p})$ and $\widetilde{\mathcal{S}}(\mathbf{p})$; however, if we do assume further smoothness, the "holey" L^∞-seminorm and true L^∞-norm can, of course, be related. For example, if we presume that $|\mathcal{S}(\mathbf{p}) - \widetilde{\mathcal{S}}(\mathbf{p})|$ has sufficiently many derivatives in the neighborhood of a unique maximizer $\overline{\mathbf{p}}$ in the interior of $\Omega \subset \mathbb{R}^M$, then (Yeşilyurt (1995))

$$\text{(56)} \qquad \xi_{1-\varepsilon_1} \sim \|\mathcal{S}(\mathbf{p}) - \widetilde{\mathcal{S}}(\mathbf{p})\|_{L^\infty(\Omega)} - \frac{1}{2}\left\{\frac{\varepsilon_1|H|^{1/2}}{\rho(\overline{\mathbf{p}})\,\omega_M}\right\}^{2/M} \qquad \text{as } \varepsilon_1 \to 0\,.$$

Here $|H|$ refers to the absolute value of the determinant of the Hessian matrix of $|\mathcal{S}(\mathbf{p}) - \widetilde{\mathcal{S}}(\mathbf{p})|$ evaluated at $\mathbf{p} = \bar{\mathbf{p}}$, and ω_M is the volume of a hypersphere in $\mathbb{R}^M$ of unit radius ($\omega_\ell = 2\pi(\frac{\ell-2}{\ell})\omega_{\ell-2}$; $\omega_2 = \pi, \omega_1 = 2$). The expression (56) again illustrates the adverse effect of increased input-space dimension, M, on predictability and localization.

4.2.2. A posteriori guide. The distribution of Y of (35) is clearly sensitive not only to $f_{W|\mathbf{P}}$, but also to the relative magnitude and*form* of $|\mathcal{S}(\mathbf{p}) - \widetilde{\mathcal{S}}(\mathbf{p})|$. The bounds of the previous section assume the worst–case scenario, in which the noise—not the variation in $|\mathcal{S}(\mathbf{p}) - \widetilde{\mathcal{S}}(\mathbf{p})|$—selects $\hat{j}$ of (38). There are, however, instances in which the form of $|\mathcal{S}(\mathbf{p}) - \widetilde{\mathcal{S}}(\mathbf{p})|$ will preclude many validation input points from contending for the $|R_j - \widetilde{\mathcal{S}}(\mathbf{P}_j)|$ sample maximizer, thereby reducing $E_{\underline{\mathbf{P}},\underline{W}|A=1}(|W_{\hat{j}}|)$, and hence $E_{\underline{\mathbf{P}},\underline{W}|A=1}(Y)$, relative to $E_{\underline{\mathbf{P}},\underline{W}|A=1}(Z)$.

In order to better reflect the form of $|\mathcal{S}(\mathbf{p}) - \widetilde{\mathcal{S}}(\mathbf{p})|$ in our estimate of the noise contribution, we propose the simple a posteriori noise guideline,

$$\hat{\mathcal{Y}}(\underline{\mathbf{P}}, \underline{W}) = |\mathcal{S}(\mathbf{P}_{\hat{j}}) + W_{\hat{j}} - \widetilde{\mathcal{S}}(\mathbf{P}_{\hat{j}})| - |\mathcal{S}(\mathbf{P}_{\hat{j}}) - \widetilde{\mathcal{S}}(\mathbf{P}_{\hat{j}})| \,. \tag{57}$$

Although neither $\mathcal{S}(\mathbf{P}_j)$ nor W_j are known individually, calculation of $\hat{j}$ from (38) requires only the sum $\mathcal{S}(\mathbf{P}_j) + W_j$, which is simply the noisy simulation result, R_j. It is true that the second term in (57) requires $\mathcal{S}(\mathbf{P}_{\hat{j}})$, however the relative additional computational effort required to recompute the noise for this *single* input point will, in general, be small. The desirable properties of the random variable $\hat{Y} = \hat{\mathcal{Y}}(\underline{\mathbf{P}}, \underline{W})$ follow directly from (39):

$$Y \leq \hat{Y} \,, \tag{58}$$

$$E_{\underline{\mathbf{P}},\underline{W}|A=1}(\hat{Y}) \leq E_{\underline{\mathbf{P}},\underline{W}|A=1}(|W_{\hat{j}}|) \leq E_{\underline{\mathbf{P}},\underline{W}|A=1}(Z) \,. \tag{59}$$

The random variable $\hat{Y}$ is thus an upper bound for the noise effect for any particular realization which, in expectation, will be sharper than our previous estimates based on the maximum sample noise. In particular, $\hat{Y}$ will only reflect the noise from the validation subsample containing input points which are viable candidates for the $|R_j - \widetilde{\mathcal{S}}(\mathbf{P}_j)|$ sample maximizer. In actual practice (see §5.2), we will construct confidence intervals for $\hat{y}$, a particular realization of our noise bound.

5. Examples.

In this section we apply our surrogate-validation algorithm and associated error analysis to several test (§5.1) and actual (§5.2) noisy simulations in order to illustrate the performance and capability of the approach. We consider only unbounded noise, and set $\alpha = 1/2$ in all examples.

5.1. Test cases. In test case I we specify $M = 1 (\mathbf{p} = p)$, $\Omega = [0, 1]$, $\rho(\mathbf{p}) = 1$, and

$$(\mathcal{S}(\mathbf{p}) - \widetilde{\mathcal{S}}(\mathbf{p}))_{\mathrm{I}} = \begin{cases} 0, & 0. \leq p < .9 \\ 1, & .9 \leq p \leq 1. \end{cases} \tag{60}$$

We take $f_{W|\mathbf{P}}$ to be normal with constant standard deviation, $\sigma_W(\mathbf{p}) = \sigma_W$. As our first set of parameters (test case I_1), we choose $\sigma_W = .1$, and $\varepsilon_1 = .1, \varepsilon_2 = .1$ (thus $N = 45$ from (33)). We then perform the validation algorithm described in §3.1 $L = 10,000$ times, and denote the validation success indicator variable, model prediction error estimate, actual noise contribution, and a posteriori noise guideline for the ℓth trial as a_ℓ, u_ℓ, y_ℓ, and $\hat{y}_\ell$, respectively. We define the number of successful validations as $L_A = \sum_{\ell=1}^{L} a_\ell$. For this test case we find $L_A/L = .902$; as expected, L_A/L is quite close to the success rate predicted from (14), $1 - \varepsilon_2 = .9$. Turning now to the noise contribution to the model prediction error estimate, we obtain $\frac{1}{L_A}\sum_{\ell=1}^{L} y_\ell a_\ell = .118$ for the empirical average of Y conditioned on $A = 1$. Our two bound procedures, (41), (46), (47) and (41), (50), (51)–(52) yield .745 and .275, respectively.

Given the form of $|\mathcal{S}(\mathbf{p}) - \widetilde{\mathcal{S}}(\mathbf{p})|$ and the relatively low noise level, both bounds are, not surprisingly, quite conservative. Conversely, $\hat{Y}$ should be extremely sharp: indeed, we find $\frac{1}{L_A}\sum_{\ell=1}^{L} \hat{y}_\ell a_\ell = .118$ for the empirical average of $\hat{Y}$ conditioned on $A = 1$.

For our second set of parameters (test case I$_2$), we select $\sigma_W = .1$ and $\varepsilon_1 = .02$, $\varepsilon_2 = .1$ (thus $N = 230$ from (33)). We now find that $L_A/L = 1.000$, significantly greater than $1 - \varepsilon_2$; since $\beta > 1$ (see (21)), our probability statement (26) is conservative. We obtain $\frac{1}{L_A}\sum_{\ell=1}^{L} y_\ell a_\ell = .192$ for the empirical average of Y conditioned on $A = 1$; the noise is larger for $\varepsilon_1 = .02$ than for $\varepsilon_1 = .1$ as N (and the validation subsample of candidate maximizers) is larger in the former case. Our two bound procedures, (41), (46), (47) and (41), (50), (51)–(52) yield 1.69 and .334, respectively. The (41), (46), (47) bound is relatively worse for $\varepsilon_1 = .02$ than for $\varepsilon_1 = .1$ because of the increased validation sample size, N, required in the $\varepsilon_1 = .02$ case; the (41), (50), (51)–(52) bound is relatively *sharper* for $\varepsilon_1 = .02$ than for $\varepsilon_1 = .1$ because the $\varepsilon_1 = .02$ case is closer to the assumed-worst-case scenario. Finally, our a posteriori noise bound, $\hat{Y}$, remains the best estimate for the noise effect, as $\hat{Y}$ correctly disregards those input sample points ($\mathbf{P}_j < .9$) which, for $\sigma_W = .1$, are not candidates for the $|R_j - \widetilde{\mathcal{S}}(\mathbf{P}_j)|$ sample maximizer: we find $\frac{1}{L_A}\sum_{\ell=1}^{L} \hat{y}_\ell a_\ell = .192$ for the empirical average of $\hat{Y}$ conditioned on $A = 1$.

For our third set of parameters (test case I$_3$) we choose $\sigma_W = 10$ and $\varepsilon_1 = .1$, $\varepsilon_2 = .1$ (thus $N = 45$ from (33)). We find $L_A/L = 1.000$, again significantly greater than $1 - \varepsilon_2$; in this case, the increased validation success rate can be traced to the bounds (27)–(29). We obtain $\frac{1}{L_A}\sum_{\ell=1}^{L} y_\ell a_\ell = 23.8$ for the empirical average of Y conditioned on $A = 1$. Our two bound procedures (41), (46), (47) and (41), (50), (51)–(52) yield 74.5 and 27.5, respectively; as expected, the latter bound is now quite sharp, since for $\sigma_W = 10$ the maximum is entirely variance-dominated. The a posteriori noise guideline can provide relatively little improvement in this case: we find $\frac{1}{L_A}\sum_{\ell=1}^{L} \hat{y}_\ell a_\ell = 24.7$ for the empirical average of $\hat{Y}$ conditioned on $A = 1$.

In test case II we specify $M = 2(\mathbf{p} = (p_{(1)}, p_{(2)}))$, $\Omega = [0, 1]^2$, $\rho(\mathbf{p}) = 1$, and

$$(\mathcal{S}(\mathbf{p}) - \widetilde{\mathcal{S}}(\mathbf{p}))_{\mathrm{II}} = \sin(\pi p_{(1)}) \sin(\pi p_{(2)}) \;. \tag{61}$$

We take $f_{W|\mathbf{P}}$ to be normal with constant standard deviation, $\sigma_W(\mathbf{p}) = \sigma_W = .1$, and choose $\varepsilon_1 = .1$, $\varepsilon_2 = .1$ (thus $N = 45$ from (33)). Performing $L = 10,000$ trials of the validation algorithm, we obtain $L_A/L = .994$; here again, the increased validation success rate originates in the bounds (27)–(29). Note that successful validation ($a_\ell = 1$) implies that $|(\mathcal{S}(\mathbf{p}) - \widetilde{\mathcal{S}}(\mathbf{p}))_{\mathrm{II}}| > u_\ell$ over an "uncharacterized" region of relative volume, μ, no greater than ε_1; for this two-dimensional input space, however, the "uncharacterized" region will, perforce, include excursions of greater than $\sqrt{\mu}$ in at least one input variable. We find $\frac{1}{L_A}\sum_{\ell=1}^{L} y_\ell a_\ell = .0736$ for the empirical average of Y conditioned on $A = 1$. Our two bound procedures (41), (46), (47) and (41), (50), (51)–(52) yield .745 and .275, respectively; as for test case I$_1$, these bounds are rather pessimistic. The a posteriori noise guideline remains relatively sharp, $\frac{1}{L_A}\sum_{\ell=1}^{L} \hat{y}_\ell a_\ell = .114$, however there is some deterioration relative to the ideal performance obtained in test case I$_1$.

5.2. Multicomponent media.

5.2.1. Formulation. We consider here the calculation of the transverse effective thermal conductivity of a random fibrous composite comprising a continuous phase of thermal conductivity unity and a dispersed phase of volumetric concentration ϕ consisting of co-oriented insulating cylindrical rods of unity diameter (Sangani and Yao (1988); Cruz and Patera (1995); Cruz, Ghaddar, and Patera (1995); Ghaddar (1995)). The calculation proceeds in a periodically replicated supercell, $\mathbf{x} \in [0, \theta] \times [0, \theta]$, containing $N_c = 4\phi\theta^2/\pi$ cylinders, as shown

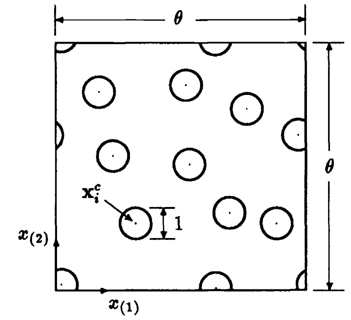


FIG. 2. *Fibrous-composite supercell* $[0,\theta]\times[0,\theta]$ *containing* N_c *cylindrical insulating inclusions.*

in Figure 2; the placement of the cylinder centers within the supercell, $\underline{\mathbf{x}}^c = (\mathbf{x}_1^c,\ldots,\mathbf{x}_{N_c}^c)$, is prescribed in terms of an assumed isotropic and homogeneous joint probability density function, $f_{\underline{\mathbf{X}}^c}(\underline{\mathbf{x}}^c;\phi,\theta)$. The (by isotropy, scalar) transverse *supercell effective conductivity*, normalized by the continuous phase conductivity, is then defined as

$$k_e(\phi,\theta) = \int_{([0,\theta]\times[0,\theta])^{N_c}} k(\underline{\mathbf{x}}^c,\phi,\theta) f_{\underline{\mathbf{X}}^c}(\underline{\mathbf{x}}^c;\phi,\theta)\underline{\mathbf{d}}\,\underline{\mathbf{x}}^c \ . \tag{62}$$

Here $k(\underline{\mathbf{x}}^c,\phi,\theta)$ is the transverse *configuration effective conductivity* for a particular θ-supercell cylinder placement, defined as the actual heat transfer across any (say) $x_{(1)}$ plane, normalized by the product of the imposed $x_{(1)}$ temperature gradient and the superficial area/depth, θ; $k(\underline{\mathbf{x}}^c,\phi,\theta)$ is a functional of the solution to the microstructure conduction problem on $\mathbf{x} = (x_{(1)}, x_{(2)}) \in [0,\theta]\times[0,\theta]$. Note that the longitudinal effective conductivity is trivial (in fact, structure-independent) for a co-oriented fibrous composite.

The effective property analysis is completed by (Cruz and Patera (1995)): defining *the* effective conductivity, $k_e^\infty(\phi)$, as $k_e(\phi,\theta\to\infty)$; determining a correlation-length function, $\Theta(\phi)$, such that, for $\theta \geq \Theta(\phi)$, (i) $k_e(\phi,\theta)$ no longer changes appreciably, and (ii) the standard deviation of the random variable K,

$$K = k(\underline{\mathbf{X}}^c,\phi,\theta) \ , \tag{63}$$

is sufficiently small. It then follows that $k_e^\infty(\phi)$ (or, equivalently, $k_e(\phi,\Theta(\phi))$ serves as an accurate effective conductivity for any macroscale problem for which the domain is large compared to $\Theta(\phi)$. (More generally, we may consider any macroscale problem for which the concentration and inclusion distribution vary only over a lengthscale large compared to $\Theta(\phi)$; within the purposiveness framework described in §2, however, we are restricted to uniform-concentration macroscale problems.) For the purposes of the current paper we will not consider the limiting process in θ; rather, we shall compute the supercell effective conductivity-concentration relationship for a particular θ, $\theta_0 = 6$, which is sufficiently large that size effects are reasonably small (Cruz and Patera (1995)). Note that, in practice, for each concentration ϕ we choose that θ closest to θ_0 for which N_c is an integer.

Finally, in order to proceed further, we must know the inclusion joint probability density function. Unfortunately, in most physical systems, details of the inclusion distribution are, at best, costly to obtain, and, more typically, simply unavailable. Furthermore, although it is possible to develop effective property *bounds* which are either independent of the inclusion joint probability density or dependent only on certain low-order correlation functions (Hashin (1983); Torquato (1991)), these bounds are often not sufficiently tight for a priori application.

For example, for our insulating fibrous medium, the best possible structure-independent *upper* bound for $k_e(\phi, \theta)$ is $\widetilde{k}_e(\phi) = (1 - \phi)/(1 + \phi)$ (Hashin (1970)), however the corresponding *lower* bound is, perforce, zero. We propose to exploit the complementary attributes of structure-dependent and structure-independent approaches by adopting the latter as surrogates for the former; if the structure-independent upper bound, $\widetilde{k}_e(\phi)$, proves an accurate estimate for $k_e(\phi, \theta_0)$ for a physically relevant inclusion density function, then (i) we will have expanded the prognostic range of the upper bound, and (ii) we will have obtained a simple, robust expression for the effective conductivity for the particular random medium of interest.

As our (arguably) physically relevant inclusion joint probability density function we assume

$$f_{\underline{\mathbf{X}}^c} = \left(\prod_{i=2}^{N_c} f_{\mathbf{x}_i^c | \mathbf{x}_{i-1}^c, \ldots, \mathbf{x}_1^c} \right) f_{\mathbf{x}_1^c} \,, \tag{64}$$

where

$$f_{\mathbf{x}_1^c}(\mathbf{x}_1^c) = 1/\theta_0^2 \,, \quad \mathbf{x}_1^c \in [0, \theta_0] \times [0, \theta_0] \,, \tag{65}$$

and

$$f_{\mathbf{x}_j^c | \mathbf{x}_{j-1}^c, \ldots, \mathbf{x}_1^c}(\mathbf{x}_j^c | \mathbf{x}_{j-1}^c, \ldots, \mathbf{x}_1^c) = \begin{cases} 0, & \mathbf{x}_j^c \notin \mathcal{P}_j \\ 1/\int_{\mathcal{P}_j} \mathbf{dx}^c, & \mathbf{x}_j^c \in \mathcal{P}_j, \end{cases} \tag{66}$$

$$\mathcal{P}_j = [0, \theta_0] \times [0, \theta_0] \setminus \bigcup_{n=1}^{j-1} D_{\mathbf{x}_n^c} \,, \tag{67}$$

for $j = 2, \ldots, N_c$. Here $D_{\mathbf{x}_n^c}$ is the (periodically extended) disk with center $\mathbf{x}_n^c$ and diameter 2. The density (64)–(67) describes a "random sequential addition" process (Widom (1966), Torquato (1991)), and is closely related to several other commonly used microstructure distributions (Torquato (1991), Cruz and Patera (1995)).

5.2.2. Simulation subproblem. Our numerical procedure consists of Monte-Carlo and finite-element treatment of the integral and integrand, respectively, of the supercell effective conductivity relationship, (62). In particular, the calculation of the configuration conductivity for a particular cylinder placement $\underline{\mathbf{X}}^c$, $K = k(\underline{\mathbf{X}}^c, \phi, \theta_0)$, is effected by (Cruz and Patera (1995); Cruz, Ghaddar, and Patera (1995); Ghaddar (1995)): automatic domain decomposition and parallel partition; automatic parallel mesh generation; second-order isoparametric finite-element discretization of the supercell conduction partial differential equation (Laplacian) over $[0, \theta_0] \times [0, \theta_0]$; parallel conjugate gradient iterative solution of the symmetric positive-definite system of finite-element equations; computation of the effective property as an energy norm on the finite-element solution; implementation on the Intel iPSC/860 hypercube multiprocessor. We occasionally encounter configurations in which extremely close inclusions either prohibit successful mesh generation, or produce discretizations characterized by excessive degrees of freedom, extreme ill-conditioning, or poor parallel load balancing. For these problematic realizations we evoke variational "nip-element" procedures which mitigate the numerical difficulties associated with geometric stiffness while simultaneously providing rigorous, *sharp* upper and lower bounds for the configuration effective conductivity (Cruz, Ghaddar, and Patera (1995)). In what follows, we shall assume that the nip-element, discretization, and incomplete-iteration errors are sufficiently small that the configuration conductivity simulation solution is effectively exact. The particular ("representative") realization shown in Figure 3 comprises 26,000 finite-element degrees of freedom and requires 2 minutes of computation on 16 nodes of the Intel iPSC/860 hypercube.

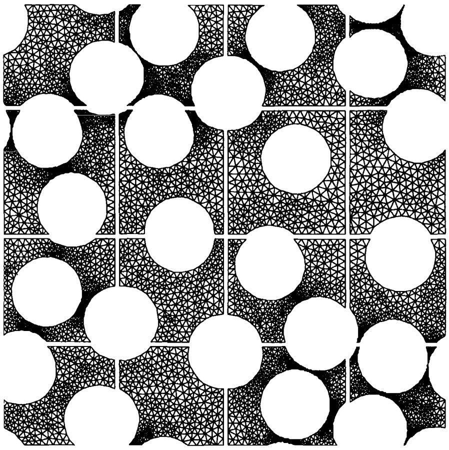


FIG. 3. *Typical geometry, parallel partition (for* 16 *processors), and finite-element mesh for inclusion concentration* $\phi = .492$.

We now turn to the Monte-Carlo component of the algorithm. For any particular concentration ϕ, we approximate the integral (62) as the sample mean

$$\overline{K} = \frac{1}{N_r} \sum_{i=1}^{N_r} K_i \,, \tag{68}$$

where $K_i = k(\underline{\mathbf{X}}_i^c, \phi, \theta_0), i = 1, \dots, N_r$. The N_r microstructure realizations, $\{\underline{\mathbf{X}}_1^c, \dots, \underline{\mathbf{X}}_{N_r}^c\}$, are generated by an acceptance–rejection Monte-Carlo sampling procedure (Rubinstein (1981)) for the joint probability density function (64)–(67). We approximate the variance of K, $\sigma_K^2(\phi, \theta_0)$, by the sample variance,

$$\sigma_K^2(\phi, \theta_0) \approx \widetilde{\sigma}_K^2(\phi, \theta_0) = \frac{1}{N_r - 1} \sum_{i=1}^{N_r} (K_i - \overline{K})^2 \,. \tag{69}$$

Finally, we assume that N_r is sufficiently large that the density of the random variable

$$V = \frac{\sqrt{N_r}[\overline{K} - k_e(\phi, \theta_0)]}{\widetilde{\sigma}_K} \tag{70}$$

is approximately normal with zero mean and unity variance.

Before turning to the validation results, we make several final remarks. First, our assumptions on V in (70) in no way affect our validation statement (14), nor, in fact, our density-independent noise bound, (41), (46), (47) (or (48)). Second, although the measurement noise is, in fact, bounded ($0 \leq K \leq \widetilde{k}_e(\phi)$), the range of $\overline{K}$ is large compared to the standard deviation of $\overline{K}$, and thus the unbounded assumption is more profitable. Third, $\sigma_K(\phi, \theta)$ will decrease with increasing θ due to spatial averaging effects. It is, however, more economical to effect variance reduction by increasing N_r than by increasing θ, as the former, first, requires less memory, and, second, leads to better conditioned problems. Nevertheless, even for $\theta \approx \Theta(\phi)$, $\sigma_K(\phi, \theta)$ will be relatively small. Fourth, for computational efficiency, N_r should be chosen so as to control—but not unnecessarily eliminate—the noise contribution to the model prediction error. We select, based on earlier "pilot" calculations (Cruz and Patera (1995)), $N_r = 20$; this modest sample size reflects the relatively small variance of K. Although for $N_r = 20$ the t-distribution is already approximately normal (Mood, Graybill, and Boes (1974)), we have no such assurances for the sample mean. Fifth, and finally, we note that, for $N_r = 20$, the effective conductivity calculation, (68), for a particular (relatively time-consuming) concentration, $\phi = .492$, requires approximately 47 minutes of computation and costs roughly \$10 on (16 processors of) the Intel iPSC/860 hypercube. If the same computation were performed on a workstation (respectively, serial/vector supercomputer), the time-to-compute (respectively, cost-to-compute) would increase significantly; the relative advantage of the parallel approach increases with increasing problem size (Fischer and Patera (1994)).

5.2.3. Surrogate validation results. We now formally identify our multicomponent surrogate problem in terms of the general variables introduced in earlier sections of the paper:

$$\begin{aligned}
M &\mapsto 1,\\
\mathbf{p} &\mapsto \phi,\\
\Omega &\mapsto [.05, .5],\\
\rho(\mathbf{p}) &\mapsto 1/.45 \text{ (uniform)},\\
\mathcal{S}(\mathbf{p}) &\mapsto k_e(\phi, \theta_0),\\
\widetilde{\mathcal{S}}(\mathbf{p}) &\mapsto \widetilde{k}_e(\phi) = (1-\phi)/(1+\phi),\\
Q &\mapsto K,\\
\sigma_Q(\mathbf{p}) &\mapsto \approx \widetilde{\sigma}_K(\phi, \theta_0),\\
\mathcal{N} &\mapsto N_r (= 20),\\
R &\mapsto \overline{K},\\
\sigma_W(\mathbf{p}) &\mapsto \approx \widetilde{\sigma}_K(\phi, \theta_0)/\sqrt{N_r},\\
g(v) &\mapsto \approx (1/2\pi)^{1/2} e^{-v^2/2}\ .
\end{aligned}$$

(The most "interesting" behavior of the effective conductivity occurs near maximum packing, $\phi \approx .82$ (Berryman (1983)). However, we do not consider this case here, as the random sequential addition Monte-Carlo sampling procedure becomes prohibitively inefficient for concentrations greater than a "jamming" threshold of roughly .55 (Sangani and Yao (1988), Torquato (1991), Cruz and Patera (1995)).) Finally, we specify $\varepsilon_1 = .1, \varepsilon_2 = .1$, and thus, from (33), $N = 45$.

Performing the validation procedure of §3.1, we find for the model prediction error estimate $u = .0574$: with confidence level greater than 90%, $|k_e(\phi, \theta_0) - \widetilde{k}_e(\phi)|$ is less than .0574 over more than 90% of the concentration range [.05, .5]. In Figure 4 we plot the surrogate-simulation discrepancy, $|\overline{K}_j - \widetilde{k}_e(\phi_j)|$, and the estimated noise standard deviation, $\widetilde{\sigma}_K(\phi_j, \theta_0)/\sqrt{N_r}$, at the validation input points, ϕ_j, $j = 1, \ldots, N$. Our bound procedures for the noise contribution to the model prediction error, (41), (46), (48) and (41), (50), (53)–(54), then yield .0273 and .0110, respectively; the latter suggests that the noise contribution to the prediction error is, relatively, not too large. Lastly, with 160 additional evaluations of K we can construct 90% confidence intervals for $k_e(\phi_{\hat{j}}, \theta_0)$, $|k_e(\phi_{\hat{j}}, \theta_0) - \widetilde{k}_e(\phi_{\hat{j}})|$, and hence $\hat{y}$ of [.290, .294], [.0462, .0507], and [.00663, .0111], respectively. This a posteriori result is ostensibly disappointing; note, however, that from (41), (50), and (53)–(54) we can only infer that $E_{\underline{\mathbf{P}},\underline{W}|A=1}(Y) \leq .0110$, whereas from (58) we can conclude (with confidence level .9) that $y \leq .0111$ for our *particular* validation realization. The a posteriori noise estimate is "sharper" than the corresponding a priori bounds because $|k_e(\phi, \theta_0) - \widetilde{k}_e(\phi)|$ is nonuniform, with significantly larger discrepancies at larger concentrations.

Finally, we briefly reconsider the optimization-purposive a posteriori analysis summarized in §2 (Yeşilyurt and Patera (1995), Yeşilyurt (1995)). We take the target value for the nondimensional effective conductivity to be $\lambda = .5$, for which our surrogate, $\widetilde{k}_e(\phi) = (1-\phi)/(1+\phi)$,

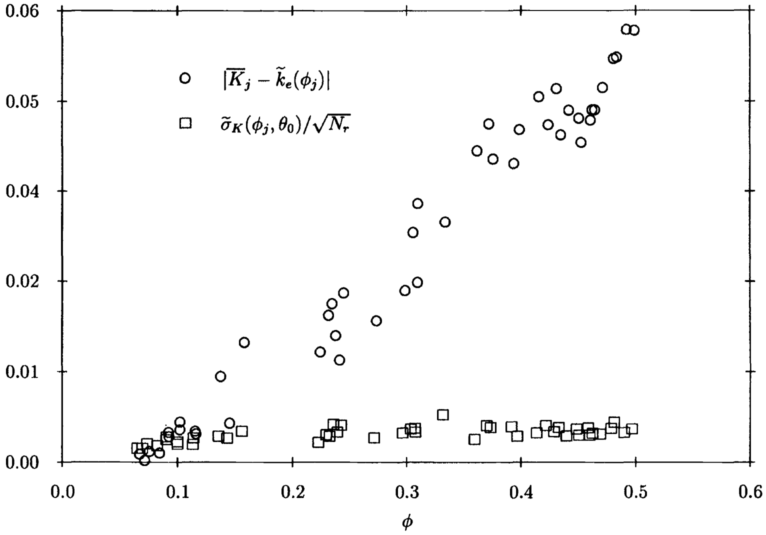


FIG. 4. *Surrogate-simulation discrepancy,* $(|\overline{K}_j - \widetilde{k}_e(\phi_j)| - \circ)$, *and estimated noise standard deviation,* $(\widetilde{\sigma}_K(\phi_j, \theta_0)/\sqrt{N_r} - \square)$, *at validation input points,* ϕ_j, $j = 1, \ldots, N$.

predicts the design point $\widetilde{\phi}^* = .333$, that is, $\widetilde{k}_e(\widetilde{\phi}^*) = .5$. Then, from (5) and (6), we can construct design intervals, $\mathcal{R}$, which contain concentration values, ϕ, for which the *actual* effective conductivity, $k_e(\phi, \theta_0)$, is within a known tolerance of λ; for example, with confidence level greater than $1-\varepsilon_2 = .9$, on over half the interval $\phi \in \mathcal{R} = [.333-.45\varepsilon_1, .333+.45\varepsilon_1] = [.288, .378]$, the actual effective conductivity, $k_e(\phi, \theta_0)$, is within $u+\delta_{\mathcal{R}} = .110\,(\delta_{\mathcal{R}} = .0528)$ of the target value $\lambda = .5$. In the context of a preliminary design exercise, this result may be sufficient without any further analysis. If we make the additional assumption that $|k_e(\phi, \theta_0) - \lambda|$ and $|\widetilde{k}_e(\phi) - \lambda|$ are quasi-convex (probably true for this apparently monotone input–output function), we can show that, with confidence level greater than $1 - \varepsilon_2 = .9$, the interval $[.155, .5]$ contains ϕ^*, the value of the concentration at which the *actual* (noise-free simulation) effective conductivity, $k_e(\phi, \theta_0)$, is closest to the target value, $\lambda = .5$ (Yeşilyurt (1995)). This (admittedly not so) restricted region can now serve, for example, as a reduced search domain in a subsequent, more refined, design exercise. Perhaps most importantly, we can now repeat our optimization analysis for different λ *without* re-appeal to the originating Monte-Carlo finite-element simulation; our probability estimates are joint over multiple studies (Yeşilyurt and Patera (1995), Yeşilyurt (1995), Otto et al. (1995)).

For the particular example presented, in which the input space is only one-dimensional, and the input–output function is rather simple, the complete surrogate framework is, arguably, an algorithmic indulgence. We contend, however, that this example is representative of more complex problems (e.g., prediction of the sedimentation rate of dense suspensions from Monte-Carlo finite-element simulations, prediction of the properties of new materials from atomistic calculations) for which the surrogate procedure is no longer a luxury, but a necessity. These problems, in turn, require improvements in, and enhancements to, the simple (train) test surrogate framework presented in this paper. Several of these new developments, including elemental approaches, sequential techniques, elemental-sequential (adaptive) procedures, proximal-candidate a posteriori analyses, and cross-validation schemes, are described in Otto et al. (1995).

Acknowledgments. We thank Professor Ben Fitzpatrick of North Carolina State University for helpful comments.

REFERENCES

B. C. Arnold and N. Balakrishnan, *Relations, Bounds, and Approximations for Order Statistics*, Lecture Notes in Statistics, 53, Springer-Verlag, Berlin, 1989.

J. G. Berryman, *Random close packing of hard spheres and disks*, Phys. Rev. A, 27 (1983), pp. 1053–1061.

G. E. P. Box, W. G. Hunter, and J. S. Hunter, *Statistics for Experimenters*, John Wiley, New York, 1978.

D. D. Cox and S. John, *A Statistical Method for Global Optimization*, Technical Report 67, Department of Statistics, University of Illinois, Urbana-Champaigne, 1992.

M. E. Cruz, C. K. Ghaddar, and A. T. Patera, *A variational-bound nip-element method for geometrically stiff problems; Application to thermal composites and porous media*, Proc. Roy. Soc. London Ser. A, 449 (1995), pp. 93–122.

M. E. Cruz and A. T. Patera, *A parallel Monte-Carlo finite-element procedure for the analysis of multicomponent random media*, Internat. J. Num. Methods Engrg., 38 (1995), pp. 1087–1121.

H. A. David, *Order Statistics*, 2nd ed., John Wiley, New York, 1981.

P. F. Fischer and A. T. Patera, *Parallel simulation of viscous incompressible flows*, Ann. Rev. Fluid Mech., 26 (1994), pp. 483–527.

G. C. Fox, M. Johnson, G. A. Lyzenga, S. W. Otto, J. K. Salmon, and D. W. Walker, *Solving Problems on Concurrent Processors: General Techniques and Regular Problems, Volume* 1, Prentice–Hall, Englewood Cliffs, NJ, 1988.

S. I. Gallant, *A connectionist learning algorithm with provable generalization and scaling bounds*, Neural Networks, 3 (1990), pp. 191–201.

C. K. Ghaddar, *Parallel Analytico-Computational Methods for Multicomponent Media; Application to Thermal Composites and Porous-Media Flows*, Ph.D. thesis, Dept. of Mech. Engrg., MIT, Cambridge, MA 1995.

J. L. Gustafson, G. R. Montry, and R. E. Benner, *Development of parallel methods for a* 1024*-processor hypercube*, SIAM J. Sci. Stat. Comput., 9 (1988), pp. 609–638.

W. Härdle, *Applied Nonparametric Regression*, Cambridge University Press, Cambridge, UK, 1990.

Z. Hashin, *Theory of composite materials*, in Mechanics of Composite Materials, F. W. Wendt, H. Liebowitz, and N. Perrone, eds., Pergamon Press, New York, 1970, pp. 201–242.

———, *Analysis of composite materials—a survey*, Trans. ASME J. Appl. Mech., 50 (1983), pp. 481–505.

Y. C. Ho, R. S. Sreenivas, and P. Vakili, *Ordinal optimization of DEDS*, Discrete Event Dynamic Systems, 2 (1992), pp. 61–88.

M. D. McKay, R. J. Beckman, and W. J. Conover, *A comparison of three methods for selecting values of input variables in the analysis of output from a computer code*, Technometrics, 21 (1979), pp. 239–245.

A. M. Mood, F. A. Graybill, and D. C. Boes, *Introduction to the Theory of Statistics*, 3rd ed., McGraw–Hill, New York, 1974.

M. D. Morris, T. J. Mitchell, and D. Ylvisaker, *Bayesian design and analysis of computer experiments: Use of derivatives in surface prediction*, Technometrics, 35 (1993), pp. 243–255.

J. Otto, Ph.D. Thesis, Dept. of Aero. and Astro., M.I.T., Cambridge, MA, in progress.

J. Otto, M. Paraschivoiu, S. Yeşİlyurt, and A. T. Patera, *Bayesian-validated computer-simulation surrogates for optimization and design*, in Proc. ICASE Workshop on Multidisciplinary Optimization, March 1995, Hampton, VA, N. Alexandrov, ed., Society for Industrial and Applied Mathematics, Philadelphia, PA.

M. Paraschivoiu, Ph.D. thesis, Dept. of Mech. Engrg., M.I.T., Cambridge, MA, in progress.

R. Y. Rubinstein, *Simulation and the Monte-Carlo Method*, John Wiley, New York, 1981.

J. Sacks, S. B. Schiller, and W. J. Welch, *Design for computer experiments*, Technometrics, 31 (1989), pp. 41–47.

J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, *Design and analysis of computer experiments*, Statist. Sci., 4 (1989), pp. 409–435.

A. S. Sangani and C. Yao, *Transport processes in random arrays of cylinders.* I. *Thermal conduction*, Phys. Fluids, 31 (1988), pp. 2426–2434.

G. A. F. Seber and C. J. Wild, *Nonlinear Regression*, John Wiley, New York, 1989.

B. W. Silverman, *Some aspects of the spline smoothing approach to non-parametric regression curve fitting*, J. Roy. Statist. Soc. B, 47 (1985), pp. 1–52.

S. Torquato, *Random heterogeneous media: Microstructure and improved bounds on effective properties*, Appl. Mech. Rev., 44 (1991), pp. 37–76.

J. F. Traub, G. W. Wasilkowski, and H. Wozniakowski, *Information-Based Complexity*, Academic Press, San Diego, 1988.

L. G. Valiant, *A theory for the learnable*, Comm. ACM, 27 (1984), pp. 1134–1142.

G. Wahba, *Smoothing noisy data by spline functions*, Numer. Math., 24 (1975), pp. 383–393.

W. J. Welch, R. J. Buck, J. Sacks, H. P. Wynn, T. J. Mitchell, and M. D. Morris, *Screening, predicting, and computer experiments*, Technometrics, 34 (1992), pp. 15–25.

B. Widom, *Random sequential addition of hard spheres to a volume*, J. Chem. Phys., 44 (1966), pp. 3888–3894.

S. Yeşİlyurt, *Construction and Validation of Computer-Simulation Surrogates for Engineering Design and Optimization*, Ph.D. thesis, Dept. of Nucl. Engrg., M.I.T., Cambridge, MA, 1995.

S. Yeşİlyurt and A. T. Patera, *Surrogates for numerical simulations; Optimization of eddy-promoter heat exchangers*, Comp. Meth. Appl. Mech. Engr., 121 (1995), pp. 231–257.

SIAM J. SCI. COMPUT.
Vol. 17, No. 4, pp. 993–1012, July 1996

013

PRIMAL-DUAL COMBINATORIAL RELAXATION ALGORITHMS FOR THE MAXIMUM DEGREE OF SUBDETERMINANTS*

SATORU IWATA[†], KAZUO MUROTA[†], AND IZUMI SAKUTA[‡]

Abstract. A primal-dual framework of combinatorial relaxation algorithms is proposed for computing the highest degree of a minor of order k of a rational function matrix. The algorithm can be used for computing the index of nilpotency of a matrix pencil (or the index of the associated differential algebraic equation). It is a linear algebraic version of the Hungarian method for the assignment problem. The proposed framework stands in contrast to the previous combinatorial relaxation algorithm based on weighted matchings, and may also be regarded as an extension of the Wolovich algorithm for row/column properness. Several algorithms are evaluated through computer experiments.

Key words. combinatorial relaxation, degree of subdeterminant, index of DAE, Kronecker form, Smith–McMillan form at infinity

AMS subject classifications. 15A15, 15A22, 05C50, 68Q25, 65L05

1. Introduction.

Let $A(x) = (A_{ij}(x))$ be an $m \times n$ rational function matrix with $A_{ij}(x)$ being a rational function in x with coefficients from the real number field $\mathbf{R}$. This paper deals with algorithms for computing the highest degree of a minor of order k ($\leq \min(m, n)$) of $A(x)$:

$$\delta_k(A) = \max_{I,J}\{\deg \det A[I, J] \mid |I| = |J| = k\}, \tag{1.1}$$

where $A[I, J]$ denotes the submatrix of A with row-set I and column-set J and the degree of a rational function $f(x) = p(x)/q(x)$ is defined by $\deg f(x) = \deg p(x) - \deg q(x)$. By convention we put $\deg f(x) = -\infty$ if $f(x) = 0$. This is one of the fundamental problems in combinatorial matrix theory [3] and has various applications in practice.

By computing $\delta_k(A)$ ($k = 1, 2, \ldots$) we can obtain the Smith–McMillan form at infinity (described in §2), which is also known as the structure at infinity in the literature of control theory (Commault–Dion [5], Hautus [12], Verghese–Kailath [23]). When $A(x)$ is a regular pencil of order n, on the other hand, $\delta_k(A)$ ($k = 1, 2, \ldots$) determines the structural indices of its Kronecker form (see Murota [15, §2.2]), and hence the index ν as well as the dynamical degree of freedom of the associated differential algebraic equations (DAEs) (Brenan–Campbell–Petzold [2], Gear [9, 10], Hairer–Wanner [11]). To be specific, the index is given by $\nu = \delta_{n-1}(A) - \delta_n(A) + 1$, whereas the dynamical degree of freedom is equal to $\delta_n(A)$. Thus the problem of computing $\delta_k(A)$ ($k = 1, 2, \ldots$) is of fundamental engineering significance.

It is not a coincidence but a necessity that the structural approach has been made recently both in the control theory and in the literature of DAEs. Taking advantage of the combinatorial nature inherent in the computation of $\delta_k(A)$ ($k = 1, 2, \ldots$) the structural approach derives combinatorial characterizations under certain genericity assumptions on the numerical values involved. In the literature of control theory a graph-theoretic method for computing the Smith–McMillan form at infinity has been developed independently by Commault, Dion, and Perez [6], Suda, Wan, and Ueno [21], and van der Woude [25] based on the primitive structural model that considers all the nonzeros independent parameters. Using the more sophisticated framework of Murota [13, Chap. 4], which distinguishes between independent parameters and

*Received by the editors July 20, 1994; accepted for publication (in revised form) April 10, 1995. This work was supported by the Sumitomo Foundation.

[†]Research Institute for Mathematical Sciences, Kyoto University, Kyoto 606-01, Japan (iwata@kurims.kyoto-u.ac.jp).

[‡]Department of Mathematical Engineering and Information Physics, University of Tokyo, Tokyo 113, Japan.

nonzero fixed constants, Murota and van der Woude [17] have derived another combinatorial characterization in terms of matroid-theoretic concepts and algorithms. For the numerical analysis of DAEs, on the other hand, extensive study has been done on the index problem of DAEs and a number of practical methods for the structural analysis have been developed (Bujakiewicz [4], Duff–Gear [7], Pantelides [19], Ungar–Kröner–Marquardt [22]).

It has been recognized through these studies that structural considerations should be useful and effective in practice and that the generic values computed by structural algorithms will have practical significance. At the same time, however, the limitation of the structural approach has also been demonstrated. A structural algorithm may fail to render the correct answer if numerical cancellations do occur for some reason or other. In emphasizing the need for an appropriate genericity assumption, Murota and van der Woude [17] demonstrated this for the Smith–McMillan form at infinity. Pantelides [19] had already recognized this phenomenon in the structural analysis of DAEs, and very recently Ungar, Kröner, and Marquardt [22] expounded upon this point, referring to the following example (significant in chemical engineering) due to Gani and Cameron [8].

The problem arises from an analysis of distillation columns and amounts to finding the index of the following matrix pencil $A(x)$ of order 13:

$$A(x) = \begin{array}{c} \\ (34) \\ (35.1) \\ (35.2) \\ (36) \\ (39) \\ (40) \\ (41) \\ (42) \\ (44.1) \\ (44.2) \\ (45.1) \\ (45.2) \\ (46) \end{array}
\begin{array}{c}
\begin{array}{ccccccccccccc} M & T & p & H^L & H^V & L & x_1 & x_2 & y_1 & y_2 & V & v_1 & v_2 \end{array} \\
\left(\begin{array}{ccccccccccccc}
x & & & & & \ast & & & & & \ast & & \\
 & & & & & \ast & x & & \ast & & \ast & & \\
 & & & & & \ast & & x & & \ast & \ast & & \\
x & & & x & \ast & \ast & & & & & \ast & & \\
\ast & & & & & \ast & & & & & & & \\
 & \ast & & \ast & & & \ast & \ast & & & & & \\
 & \ast & \ast & & \ast & & & & \ast & \ast & & & \\
\ast & & \ast & & & & & & & & & & \\
 & & & & & & & & \ast & & \ast & \ast & \\
 & & & & & & & & & \ast & \ast & & \ast \\
 & \ast & \ast & & & & \ast & \ast & & & -a_1 & 1-b_{11} & -b_{12} \\
 & \ast & \ast & & & & \ast & \ast & & & -a_2 & -b_{21} & 1-b_{22} \\
 & & & & & & & & & & 1 & -1 & -1
\end{array}\right)
\end{array},$$

where we assume $NC = 2$ in the notation of [22], and attach the equation numbers and the variable names taken from [22]. The symbol $\ast$, designating an occurrence of a variable in an equation, is to be considered an independent (or generic) parameter. In contrast, the six parameters $a_1, a_2, b_{11}, b_{21}, b_{12}, b_{22}$ appearing in the lower-right corner cannot be regarded as independent parameters, since they satisfy the relations

$$a_1 + a_2 = 1, \quad b_{11} + b_{21} = 0, \quad b_{12} + b_{22} = 0.$$

Structural algorithms, ignoring these relations among the parameters, yield an erroneous answer $\widehat{\nu} = 1$ in contradiction to the true index $\nu = 2$.

In this paper we are interested in algorithms which yield correct values even in the nongeneric case, namely those algorithms which return $\nu = 2$ for the above example. In particular, we are interested in such algorithms that rely primarily on structural information and ask for numerical information only when it is needed to ensure correct answers. The recent algorithm of Murota [15] is of this kind. It is a combinatorial relaxation-type algorithm that employs the weighted matching problem as a relaxed problem. In contrast, the elimination-type algorithm (described in §2) does not fall into this category, though it will be considered for comparison in our evaluation of the algorithms.

An algorithm of combinatorial relaxation type generally consists of the following three phases.

Phase 1: Consider a relaxed (or an easier) combinatorial problem of the original algebraic problem and find a solution to the relaxed one.

Phase 2: Test for the validity of this solution with respect to the original problem. If valid, output the solution and stop.

Phase 3 (in case of invalid solution): Modify the relaxation so that the invalid solution is eliminated.

This type of algorithm has been found for the Newton polygon for Puiseux-series solutions to determinantal equations [14] and for the degree of determinant of a skew-symmetric matrix [16].

It may be said that such an approach was already implicit in the algorithm described in Wolovich [24] for transforming a polynomial matrix into the row-proper form and for computing δ_m as a byproduct, where m denotes the number of rows. This algorithm takes the sum of row degrees (the maximum degrees of the entries in a row) as an estimate of δ_m and tests for row properness. In case the estimate turns out to be nonproper it modifies the matrix so as to improve the estimate without changing δ_m. From this, we might regard it as a combinatorial relaxation-type algorithm. We extend this algorithm for computing $\delta_k(A)$ for $k = 1, \dots, r$, where r denotes the rank of A.

The present paper aims to propose a primal-dual framework of combinatorial relaxation algorithms for computing the maximum degree of subdeterminants. Like the Hungarian method for the assignment problem [1, 20], the algorithms keep a set of admissible potentials (feasible dual variables), which determines the estimate (dual objective value), and check its tightness (dual optimality) by solving a simpler (restricted primal) problem. If it turns out to be nontight, we obtain information useful for modifying the potentials as well as the matrix. Thus the proposed framework can be interpreted as a linear algebraic version of the Hungarian method. It stands in contrast to the previous combinatorial relaxation algorithm based on weighted matchings (see Remarks 3.1 and 3.2) and may also be regarded as an extension of the Wolovich algorithm for row/column properness. Several algorithms are evaluated through computer experiments.

This paper is organized as follows. In §2, we describe the Smith–McMillan form at infinity and its relationship to the maximum degree of subdeterminants. Section 3 provides the framework, the details, and the time complexity of the proposed algorithm. Experimental results are described in §4, followed by discussions in §5.

2. Smith–McMillan form at infinity.

In this section, we describe the Smith–McMillan form at infinity, which is a canonical form of a rational function matrix. A rational function $f(x)$ is called proper if $\deg f(x) \leq 0$, and a rational function matrix $A(x)$ is also called proper if all of its entries are proper. A nonsingular proper matrix $U(x)$ is said to be biproper if its inverse is also proper, or equivalently, if $\deg \det U(x) = 0$. The Smith–McMillan form at infinity is the canonical form of a rational function matrix under biproper equivalency as stated below (Hautus [12], Verghese–Kailath [23]).

THEOREM 2.1 (Smith–McMillan form at infinity). *For any rational function matrix $A(x)$ of rank r, there exist biproper matrices $U(x)$ and $V(x)$ such that*

$$U(x)A(x)V(x) = \begin{pmatrix} \Gamma(x) & O \\ O & O \end{pmatrix}, \tag{2.1}$$

where

$$\Gamma(x) = \operatorname{diag}(x^{t_1}, x^{t_2}, \dots, x^{t_r}), \quad t_1 \geq t_2 \geq \cdots \geq t_r.$$

Since the maximum degrees of subdeterminants δ_k $(k = 1, \ldots, r)$ defined in (1.1) are invariant under biproper transformations, we have

$$t_k = \delta_k(A) - \delta_{k-1}(A) \quad (k = 1, \ldots, r),$$

where $\delta_0(A) = 0$ by convention. Therefore we can compute the set of δ_k $(k = 1, \ldots, r)$ from the Smith–McMillan form at infinity, and conversely, the latter from the former.

We note that the set of proper rational functions forms a Euclidean ring with a rather trivial division rule. By specializing the standard elimination procedure in a general Euclidean ring [18], we can obtain a procedure for the Smith form in the proper rational function ring, and furthermore a procedure for the Smith–McMillan form at infinity as a minor variant thereof.

Elimination Algorithm for Smith–McMillan Form at Infinity.

Step 1: If $A(x) = 0$, then go to Step 6.

Step 2: Permute rows and columns independently so that $\deg A_{11}(x)$ is larger than or equal to that of any other entry.

Step 3: For each entry in the first row or in the first column, calculate its quotient:

$$q_{i1}(x) = \frac{A_{i1}(x)}{A_{11}(x)} \quad \text{for } i = 2, 3, \ldots, m,$$

$$q_{1j}(x) = \frac{A_{1j}(x)}{A_{11}(x)} \quad \text{for } j = 2, 3, \ldots, n.$$

Step 4: Subtract (the first row)×$q_{i1}(x)$ from the ith row $(i = 2, \ldots, m)$, and then subtract (the first column)×$q_{1j}(x)$ from the jth column $(j = 2, \ldots, n)$. Then we obtain a matrix as follows:

$$\left(\begin{array}{c|ccc} A_{11}(x) & 0 & \cdots & 0 \\ \hline 0 & A_{22}(x) & \cdots & A_{2n}(x) \\ \vdots & \vdots & \ddots & \vdots \\ 0 & A_{m2}(x) & \cdots & A_{mn}(x) \end{array}\right).$$

Note that $\deg A_{11}(x) \geq \deg A_{ij}(x)$ $(i = 2, \ldots, m; j = 2, \ldots, n)$.

Step 5: Execute Step 1 to Step 4 on the lower-right submatrix recursively.

Step 6: At this point, $A(x)$ has been transformed to

$$\left(\begin{array}{cccc|c} A_{11}(x) & 0 & \cdots & 0 & \\ 0 & A_{22}(x) & \ddots & \vdots & O \\ \vdots & \ddots & \ddots & 0 & \\ 0 & \cdots & 0 & A_{rr}(x) & \\ \hline & & O & & O \end{array}\right),$$

where $\deg A_{11}(x) \geq \deg A_{22}(x) \geq \cdots \geq \deg A_{rr}(x)$. We can further transform it to the form (2.1) with $t_k = \deg A_{kk}(x)$ for $k = 1, \ldots, r$ by multiplying the kth row with $x^{t_k}/A_{kk}(x)$ for $k = 1, \ldots, r$.

The elimination procedure above requires $O(\min(m, n)mn)$ additions and multiplications of rational functions. After several iterations, however, the number of terms in each entry will increase. Therefore, the whole procedure takes more than $O(\min(m, n)mn)$ time. This phenomenon is shown in §4.

3. Combinatorial relaxation algorithms.

In this section, we present a new framework of combinatorial relaxation-type algorithms for computing δ_k. This shares the approach of "combinatorial relaxation" with the previous algorithm of Murota [15], while featuring potentials (dual variables) rather than matchings in constructing the combinatorial estimate. The proposed framework may be thought of as a linear algebraic version of the primal-dual approach to the assignment problem that is well known in the context of combinatorial optimization [1, 20].

3.1. Outline of the proposed framework.

For a rational function matrix $A(x)$, let R and C denote the row-set and the column-set, respectively. By potentials we mean integer numbers associated with rows or columns. With appropriately chosen potentials, say, $\rho = (\rho_i \mid i \in R)$ and $\gamma = (\gamma_j \mid j \in C)$, we compute an estimate $\widehat{\delta_k}(A; \rho, \gamma)$ of $\delta_k(A)$ such that $\delta_k(A) \leq \widehat{\delta_k}(A; \rho, \gamma)$. We then test for tightness (equality) in this inequality without computing $\delta_k(A)$. If the estimate turns out to be nontight, we modify the matrix $A(x)$ as well as the potentials.

The outline of the combinatorial relaxation algorithm to compute δ_k for a fixed k is summarized as follows (see §3.5 for a concrete example).

Phase 0: Choose appropriate (ρ, γ).

Phase 1: Compute the combinatorial upper estimate $\widehat{\delta_k}(A; \rho, \gamma)$.

Phase 2: Test whether $\widehat{\delta_k}(A; \rho, \gamma) = \delta_k(A)$ or not. If the equality holds, output $\widehat{\delta_k}(A; \rho, \gamma)$ and stop.

Phase 3: Modify A to another matrix A' and (ρ, γ) to (ρ', γ') such that $\delta_k(A') = \delta_k(A)$ and $\widehat{\delta_k}(A'; \rho', \gamma') < \widehat{\delta_k}(A; \rho, \gamma)$, namely A' has the same true value as A and an improved estimate. Put $A := A'$, $\rho := \rho'$, $\gamma := \gamma'$ and go to Phase 1.

3.2. Phase 1—combinatorial estimation.

The combinatorial estimate, denoted as $\widehat{\delta_k}(A; \rho, \gamma)$ in §3.1, is specified in this section. Define c_{ij} by

$$c_{ij} = \deg A_{ij}(x), \tag{3.1}$$

where by convention we put $c_{ij} = -\infty$ if $A_{ij}(x) = 0$. We choose a set of potentials $\rho = (\rho_i \mid i \in R)$ and $\gamma = (\gamma_j \mid j \in C)$ which satisfies the condition

$$\rho_i + \gamma_j \geq c_{ij} \quad \text{for } i \in R \text{ and } j \in C. \tag{3.2}$$

Such potentials are said to be admissible. Then for $I \subseteq R$ and $J \subseteq C$ with $|I| = |J|$ it holds that

$$\deg\det A[I, J] \leq \rho(I) + \gamma(J), \tag{3.3}$$

where $\rho(I) = \sum_{i\in I} \rho_i$ and $\gamma(J) = \sum_{j\in J} \gamma_j$.

By considering the maximum possible value of the right-hand side, i.e., by introducing

$$\widehat{\delta_k}(A; \rho, \gamma) = \max_{|I|=k} \rho(I) + \max_{|J|=k} \gamma(J),$$

we obviously have

$$\rho(I) + \gamma(J) \leq \widehat{\delta_k}(A; \rho, \gamma) \tag{3.4}$$

for $I \subseteq R$ and $J \subseteq C$ with $|I| = |J| = k$. The inequalities (3.3) and (3.4) imply

$$\delta_k(A) \leq \widehat{\delta_k}(A; \rho, \gamma), \tag{3.5}$$

which shows that $\widehat{\delta_k}(A;\rho,\gamma)$ serves as an upper bound on the true value. Remember that (ρ,γ) must be admissible in the sense of (3.2) before (3.5) can be true.

The estimate $\widehat{\delta_k}(A;\rho,\gamma)$ admits an alternative expression suitable for its efficient computation. Consider the following permutations for the row-set R and the column-set C of A:

$$\tau:\{1,\dots,m\}\to R \qquad \text{such that } \rho_{\tau(1)}\ge\rho_{\tau(2)}\ge\cdots\ge\rho_{\tau(m)}; \tag{3.6}$$

$$\sigma:\{1,\dots,n\}\to C \qquad \text{such that } \gamma_{\sigma(1)}\ge\gamma_{\sigma(2)}\ge\cdots\ge\gamma_{\sigma(n)}. \tag{3.7}$$

Then it is easy to see that

$$\widehat{\delta_k}(A;\rho,\gamma)=\sum_{i=1}^{k}\rho_{\tau(i)}+\sum_{j=1}^{k}\gamma_{\sigma(j)}.$$

REMARK 3.1. We could optimize the choice of potentials (ρ,γ) to minimize the estimate $\widehat{\delta_k}(A;\rho,\gamma)$. Then the estimate $\widehat{\delta_k}(A;\rho,\gamma)$ would coincide in value with the maximum weight of a matching (transversal) of size k, which has been employed in [15] as another combinatorial estimate for $\delta_k(A)$, and would satisfy (3.5) with equality in the generic case (see [15, Thm. 3]). In this paper, however, we consider algorithms that do not rely on the optimal choice of (ρ,γ).

3.3. Phase 2—test for tightness. This section provides a way to test whether $\delta_k(A)=\widehat{\delta_k}(A;\rho,\gamma)$ or not without computing $\delta_k(A)$ directly.

Define $I^\sharp$, $J^\sharp$, $\tilde I$, and $\tilde J$ as follows:

$$I^\sharp=\{i\in R\mid \rho_i>\rho_{\tau(k+1)}\},\qquad \tilde I=\{i\in R\mid \rho_i\ge\rho_{\tau(k)}\}; \tag{3.8}$$

$$J^\sharp=\{j\in C\mid \gamma_j>\gamma_{\sigma(k+1)}\},\qquad \tilde J=\{j\in C\mid \gamma_j\ge\gamma_{\sigma(k)}\}. \tag{3.9}$$

Exceptionally, in case $k=m$, we put $I^\sharp=\tilde I=R$, and in case $k=n$, we put $J^\sharp=\tilde J=C$. Then $\rho(I)$ attains the maximum subject to $|I|=k$ if and only if $I^\sharp\subseteq I\subseteq\tilde I$. Similarly, $\gamma(J)$ is maximized subject to $|J|=k$ if and only if $J^\sharp\subseteq J\subseteq\tilde J$. Therefore (3.4) holds with equality if and only if $I^\sharp\subseteq I\subseteq\tilde I$ and $J^\sharp\subseteq J\subseteq\tilde J$. Define a constant matrix $A^\sharp$ by

$$A^\sharp=(A^\sharp_{ij}),\qquad A^\sharp_{ij}=\begin{cases}\lim\limits_{x\to\infty}x^{-\rho_i-\gamma_j}A_{ij}(x) & \text{if } i\in\tilde I \text{ and } j\in\tilde J;\\ 0 & \text{otherwise.}\end{cases}$$

PROPOSITION 3.1 (tightness). *The following three conditions* (a)–(c) *are equivalent.*

(a) $\delta_k(A)=\widehat{\delta_k}(A;\rho,\gamma)$.

(b) *There exist* $I\supseteq I^\sharp$ *and* $J\supseteq J^\sharp$ *such that* $\operatorname{rank}A^\sharp[I,J]=|I|=|J|=k$.

(c) *The following four conditions* (r1)–(r4) *are satisfied:*

(r1) $\operatorname{rank}A^\sharp[R,C]\ge k$;

(r2) $\operatorname{rank}A^\sharp[I^\sharp,C]=|I^\sharp|$;

(r3) $\operatorname{rank}A^\sharp[R,J^\sharp]=|J^\sharp|$;

(r4) $\operatorname{rank}A^\sharp[I^\sharp,J^\sharp]\ge|I^\sharp|+|J^\sharp|-k$.

Proof. See Lemma 6 of [15] for (b)$\Leftrightarrow$(c). Let us prove (a)$\Leftrightarrow$(b). From the definition of $A^\sharp$, we have

$$A_{ij}(x)=x^{\rho_i+\gamma_j}(A^\sharp_{ij}+o(1))\quad \text{for } i\in\tilde I \text{ and } j\in\tilde J,$$

which implies

$$\det A[I,J]=x^{\rho(I)+\gamma(J)}(\det A^\sharp[I,J]+o(1)) \tag{3.10}$$

if $I\subseteq\tilde I$, $J\subseteq\tilde J$, and $|I|=|J|$.

Suppose $\delta_k = \widehat{\delta_k}$, and then there exist I and J with $|I| = |J| = k$ that satisfy both (3.3) and (3.4) with equalities. Hence $I \subseteq \tilde{I}$ and $J \subseteq \tilde{J}$, and it follows from (3.10) that $A^\sharp[I, J]$ is nonsingular. Thus (a)⇒(b) has been shown. Conversely, suppose $I \supseteq I^\sharp$, $J \supseteq J^\sharp$, and $\operatorname{rank} A^\sharp[I, J] = |I| = |J| = k$. Then we must have $I \subseteq \tilde{I}$ and $J \subseteq \tilde{J}$, and consequently (3.4) holds with equality. Furthermore, from (3.10), we obtain an equality in (3.3). Therefore, $\deg \det A[I, J] = \widehat{\delta_k}$, which together with (3.5) shows $\delta_k = \widehat{\delta_k}$. □

According to Proposition 3.1 above, the test for $\widehat{\delta_k} = \delta_k$ in Phase 2 can be reduced to computing the ranks of the four constant matrices. Actually, for (r1), (r2), and (r4), we use the Gaussian elimination by elementary row transformations, and for (r3), we use the Gaussian elimination by elementary column transformations.

3.4. Phase 3—modification. When the estimate $\widehat{\delta_k}(A; \rho, \gamma)$ turns out to be nontight, we modify the matrix as well as the potentials. Recall that the maximum degree of subdeterminants of a fixed order is invariant under biproper equivalence transformations. That is,

$$A'(x) = U(x)A(x)V(x) \tag{3.11}$$

satisfies $\delta_k(A') = \delta_k(A)$ if both $U(x)$ and $V(x)$ are biproper. We use this type of transformation in the modification of the matrix $A(x)$.

Let U be a nonsingular constant matrix of order m such that

$$U_{hi} = 0 \quad \text{if } \rho_h > \rho_i,$$

and define a transformation matrix $U(x)$ by

$$U(x) = \operatorname{diag}(x; \rho) \cdot U \cdot \operatorname{diag}(x; -\rho), \tag{3.12}$$

where $\operatorname{diag}(x; \rho)$ designates the diagonal matrix $\operatorname{diag}(x^{\rho_1}, x^{\rho_2}, \ldots, x^{\rho_m})$. Then $U(x)$ is a biproper matrix. Similarly, using a nonsingular constant matrix V that satisfies the condition

$$V_{jh} = 0 \quad \text{if } \gamma_h > \gamma_j,$$

we define a transformation matrix $V(x)$ by

$$V(x) = \operatorname{diag}(x; -\gamma) \cdot V \cdot \operatorname{diag}(x; \gamma), \tag{3.13}$$

which is also a biproper matrix.

We now explain how to construct such a pair of constant matrices U and V efficiently according to which of the four conditions (r1)–(r4) is violated by the constant matrix $A^\sharp$ (see Proposition 3.1 (c)).

When (r1) is violated: In checking the condition (r1) by means of the Gaussian row elimination with column pivoting, applied to the matrix $A^\sharp$ with rows rearranged by τ according to the magnitude of ρ_i, we obtain a nonsingular matrix S of order m such that $S_{\tau(l)\tau(i)} = 0$ if $l < i$ and such that $SA^\sharp$ has $(m - \operatorname{rank} A^\sharp)$ zero vectors among its row vectors. Let H be a subset of $\tilde{I}$ such that $|H| = |\tilde{I}| - k + 1$ and $(SA^\sharp)[H, C] = O$. Construct the constant matrix U as follows:

$$U[H, R] = S[H, R], \ U[R - H, H] = O, \ U[R - H, R - H] = I_{|R-H|}, \tag{3.14}$$

where $I_{|R-H|}$ denotes the unit matrix. Let V be the unit matrix of order n.

When (r2) is violated: In checking the condition (r2) by the Gaussian row elimination with column pivoting, applied to the matrix $A^\sharp[I^\sharp, C]$ with rows rearranged by τ according to the magnitude of ρ_i, we obtain a nonsingular matrix S of order $|I^\sharp|$ such that $S_{\tau(l)\tau(i)} = 0$ if $l < i$ and such that $S \cdot A^\sharp[I^\sharp, C]$ has $(|I^\sharp| - \operatorname{rank} A^\sharp[I^\sharp, C])$ zero row vectors. Let $H \subseteq I^\sharp$ be the singleton set corresponding to a zero row vector in $S \cdot A^\sharp[I^\sharp, C]$. Then construct U as in (3.14). Let V be the unit matrix of order n.

When (r3) is violated: In checking the condition (r3) by the Gaussian column elimination with row pivoting, applied to the matrix $A^{\sharp}[R, J^{\sharp}]$ with columns rearranged by σ according to the magnitude of γ_j, we obtain a nonsingular matrix S of order $|J^{\sharp}|$ such that $S_{\sigma(j)\sigma(l)} = 0$ if $l < j$ and such that $A^{\sharp}[R, J^{\sharp}] \cdot S$ has $(|J^{\sharp}| - \mathrm{rank}\, A^{\sharp}[R, J^{\sharp}])$ zero column vectors. Construct the constant matrix V of order n in a (transposed) way similar to that in which we constructed U in the case of (r2), namely

$$(3.15)\quad V[C, H] = S[C, H],\ V[H, C - H] = O,\ V[C - H, C - H] = I_{|C-H|}.$$

Let U be the unit matrix of order m.

When (r4) is violated: In checking the condition (r4) by means of the Gaussian row elimination with column pivoting, applied to the matrix $A^{\sharp}[I^{\sharp}, J^{\sharp}]$ with rows rearranged by τ according to the magnitude of ρ_i, we obtain a nonsingular matrix S of order $|I^{\sharp}|$ such that $S_{\tau(l)\tau(i)} = 0$ if $l < i$ and such that $S \cdot A^{\sharp}[I^{\sharp}, J^{\sharp}]$ has $(|I^{\sharp}| - \mathrm{rank}\, A^{\sharp}[I^{\sharp}, J^{\sharp}])$ zero row vectors. Let H be a subset of $I^{\sharp}$ corresponding to $(k - |J^{\sharp}| + 1)$ zero row vectors in $S \cdot A^{\sharp}[I^{\sharp}, J^{\sharp}]$. Then construct the constant matrix U in the same way as in (3.14). Let V be the unit matrix of order n.

After modifying the matrix $A(x)$ to $A'(x)$ by (3.11) with $U(x)$ and $V(x)$ thus constructed, the set of potentials is updated as follows. Put $c'_{ij} = \deg A'_{ij}(x)$ as in (3.1). It follows from (3.11), (3.12), and (3.13) that

$$\mathrm{diag}\,(x; -\rho) \cdot A'(x) \cdot \mathrm{diag}\,(x; -\gamma) = U \cdot \mathrm{diag}\,(x; -\rho) \cdot A(x) \cdot \mathrm{diag}\,(x; -\gamma) \cdot V,$$

which, together with (3.2), implies

$$(3.16)\qquad \rho_i + \gamma_j \geq c'_{ij} \quad \text{for } i \in R \quad \text{and } j \in C.$$

We update the set of potentials to improve the estimate, exploiting the fact that (3.16) holds with strict inequality for some $i \in R$ and $j \in C$. Roughly speaking, we reduce the potentials of such rows or columns without violating the admissibility. A more specific description, according to which condition of the four is violated, is given below.

When (r1) is violated:

$$(3.17)\ \rho'_i = \begin{cases} \max\left\{ \max\limits_{j \in \tilde{J}}(c'_{ij} - \gamma_j),\ \max\limits_{j \in (C - \tilde{J})} c'_{ij} - \gamma_{\sigma(k)} \right\} & \text{if } i \in \tilde{I}, \\ \rho_i & \text{otherwise;} \end{cases}$$

$$(3.18)\ \gamma'_j = \max_{i \in R}(c'_{ij} - \rho'_i).$$

When (r2) is violated:

$$(3.19)\ \rho'_i = \begin{cases} \max\left\{ \max\limits_{j \in \tilde{J}}(c'_{ij} - \gamma_j),\ \max\limits_{j \in (C - \tilde{J})} c'_{ij} - \gamma_{\sigma(k)} \right\} & \text{if } i \in I^{\sharp}, \\ \rho_i & \text{otherwise;} \end{cases}$$

$$(3.20)\ \gamma'_j = \max_{i \in R}(c'_{ij} - \rho'_i).$$

When (r3) is violated:

$$(3.21)\ \gamma'_j = \begin{cases} \max\left\{ \max\limits_{i \in \tilde{I}}(c'_{ij} - \rho_i),\ \max\limits_{i \in (R - \tilde{I})} c'_{ij} - \rho_{\tau(k)} \right\} & \text{if } j \in J^{\sharp}, \\ \gamma_j & \text{otherwise;} \end{cases}$$

$$(3.22)\ \rho'_i = \max_{j \in C}(c'_{ij} - \gamma'_j).$$

When (r4) is violated:

$$(3.23)\quad \rho_i' = \begin{cases} \rho_i - \beta & \text{if } i \in H; \\ \rho_i & \text{otherwise,} \end{cases} \qquad \gamma_j' = \begin{cases} \gamma_j & \text{if } j \in J^\sharp, \\ \gamma_j + \beta & \text{otherwise,} \end{cases}$$

where

$$(3.24)\quad \beta = \min \left\{ \rho_{\tau(|I^\sharp|)} - \rho_{\tau(|I^\sharp|+1)}, \quad \gamma_{\sigma(|J^\sharp|)} - \gamma_{\sigma(|J^\sharp|+1)}, \quad \min_{h \in H, j \in J^\sharp} (\rho_h + \gamma_j - c_{hj}') \right\}.$$

Exceptionally, the first term is excluded from the minimization when $I^\sharp = R$, and the second is excluded when $J^\sharp = C$.

As to the admissibility of the updated set of potentials, the following lemma holds.

LEMMA 3.2. *The updated set of potentials (ρ', γ') is admissible with respect to the modified matrix A'. That is, $\rho_i' + \gamma_j' \geq c_{ij}'$ for $i \in R$ and $j \in C$.*

Proof. The proof is immediate when (r1), (r2), or (r3) is violated. When (r4) is violated, however, this lemma follows from (3.16). □

The following theorem shows that the estimate is improved by the modification described above.

THEOREM 3.3. *The following inequality holds:*

$$(3.25)\qquad \widehat{\delta}_k(A'; \rho', \gamma') < \widehat{\delta}_k(A; \rho, \gamma).$$

Proof. With respect to ρ' and γ', we define permutations τ' of R and σ' of C, in a similar way to τ, σ (see (3.6) and (3.7)). We now prove (3.25) according to which condition of the four is violated.

When (r1) or (r2) is violated: It follows from the definition of ρ' and (3.16) that $\rho_i' \leq \rho_i$ for $i \in R$. Furthermore, since $\rho_h + \gamma_j > c_{hj}'$ for $h \in H$ and $j \in \tilde{J}$, we have $\rho_h' < \rho_h$ for $h \in H$. Similarly, we have $\gamma_j' \leq \gamma_j$ for $j \in \tilde{J}$ and $\gamma_j' \leq \gamma_{\sigma(k)}$ for $j \in (C - \tilde{J})$. As a result,

$$\sum_{i=1}^{k} \rho_{\tau'(i)}' < \sum_{i=1}^{k} \rho_{\tau(i)}, \qquad \sum_{j=1}^{k} \gamma_{\sigma'(j)}' \leq \sum_{j=1}^{k} \gamma_{\sigma(j)}.$$

Hence, $\widehat{\delta}_k(A'; \rho', \gamma') < \widehat{\delta}_k(A; \rho, \gamma)$.

When (r3) is violated: It can be shown in a way similar to that in which the case (r2) is violated.

When (r4) is violated: Since $\rho_h + \gamma_j > c_{hj}'$ for $h \in H$ and $j \in J^\sharp$, we have $\beta > 0$. It follows from the definition of ρ' that

$$\sum_{i=1}^{k} \rho_{\tau'(i)}' \leq \sum_{i=1}^{k} \rho_{\tau(i)} - \beta |H|.$$

On the other hand, the definition of γ' implies

$$\sum_{j=1}^{k} \gamma_{\sigma'(j)}' \leq \sum_{j=1}^{k} \gamma_{\sigma(j)} + \beta(k - |J^\sharp|).$$

Since $|H| = k - |J^\sharp| - 1$ and $\beta > 0$, we obtain $\widehat{\delta}_k(A'; \rho', \gamma') < \widehat{\delta}_k(A; \rho, \gamma)$. □

REMARK 3.2. The previous algorithm [15] of combinatorial relaxation type mentioned in Remark 3.1, which will be called the matching method in this paper, also makes use of the potentials (dual variables). The test for tightness is reduced to the rank conditions on a constant matrix defined with the aid of the optimal dual variables. If the combinatorial estimate (maximum weight of matchings) turns out to be nontight, the matrix is modified by a biproper transformation determined with reference to the optimal dual variables. The set of potentials, however, is not updated from the current one, but recomputed by solving the weighted matching problem from scratch. Thus the matchings, which are the combinatorial counterpart of determinants, play the primary role in the matching method. In this sense, the matching method might be regarded as a primal approach rather than a primal-dual.

3.5. Example. Let us consider $\delta_3(A)$ for a polynomial matrix

$$A(x) = \begin{array}{c} \\ \mathrm{R1} \\ \mathrm{R2} \\ \mathrm{R3} \\ \mathrm{R4} \end{array} \!\!\begin{array}{c} \begin{array}{cccc} \mathrm{C1} & \mathrm{C2} & \mathrm{C3} & \mathrm{C4} \end{array} \\ \begin{pmatrix} x^4 & x^5 & 0 & 2x^3 \\ x^5 & x^6+1 & x^4 & x^2 \\ x^4+x & x^5 & -x^3 & 0 \\ 2x^2 & x & 0 & x+2 \end{pmatrix} \end{array},$$

following Example 1 of [15]. As an initial set of admissible potentials, we take

$$\rho = (0,0,0,0), \quad \gamma = (5,6,4,3).$$

Then we have $\widehat{\delta_3}(A;\rho,\gamma) = (0+0+0)+(6+5+4) = 15$ and

$$I^\sharp = \emptyset, \quad \tilde{I} = \{\mathrm{R1},\mathrm{R2},\mathrm{R3},\mathrm{R4}\},$$
$$J^\sharp = \{\mathrm{C1},\mathrm{C2},\mathrm{C3}\}, \quad \tilde{J} = \{\mathrm{C1},\mathrm{C2},\mathrm{C3}\}.$$

Hence

$$A^\sharp = \begin{array}{c} \begin{array}{ccc} \longleftarrow & J^\sharp & \longrightarrow \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \end{array}.$$

In this case, (r1) and (r3) are violated. Let us change $A(x)$ according to the modification rule for the case when (r1) is violated (see §3.4). The modification matrix $U(x)$ is the unit matrix, and hence $A'(x) = A(x)$ (not changed). The set of potentials, however, is updated to

$$\rho' = (-1,0,-1,-3), \quad \gamma' = (5,6,4,4).$$

Then we have $\widehat{\delta_3}(A';\rho',\gamma') = (0+(-1)+(-1))+(6+5+4) = 13$ and

$$I^\sharp = \{\mathrm{R1},\mathrm{R2},\mathrm{R3}\}, \quad \tilde{I} = \{\mathrm{R1},\mathrm{R2},\mathrm{R3}\},$$
$$J^\sharp = \{\mathrm{C1},\mathrm{C2}\}, \quad \tilde{J} = \{\mathrm{C1},\mathrm{C2},\mathrm{C3},\mathrm{C4}\}.$$

Hence

$$(A')^\sharp = \begin{array}{c} \\ \uparrow \\ I^\sharp \\ \downarrow \\ \end{array} \!\!\begin{array}{c} \begin{array}{ccc} \leftarrow J^\sharp & & \longrightarrow \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 2 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \end{array}.$$

In this case, (r3) and (r4) are violated. First suppose we change $A'(x)$ according to the modification rule for the case of (r3), whereas the other possibility is considered later. We detect the violation of (r3) through the Gaussian column elimination

$$(A')^\sharp[R, J^\sharp] \cdot S = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 0 \end{pmatrix},$$

in which we note $\gamma'_{\mathrm{C2}} > \gamma'_{\mathrm{C1}}$. Hence we have $H = \{\mathrm{C1}\}$ and

$$V = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

According to (3.13) we have

$$\begin{aligned} V(x) \quad &= \mathrm{diag}\,(x^{-5}, x^{-6}, x^{-4}, x^{-4}) \cdot V \cdot \mathrm{diag}\,(x^5, x^6, x^4, x^4) \\ &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ -x^{-1} & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \end{aligned}$$

which modifies $A'(x)$ to

$$A''(x) = A'(x)V(x) = \begin{pmatrix} 0 & x^5 & 0 & 2x^3 \\ -x^{-1} & x^6+1 & x^4 & x^2 \\ x & x^5 & -x^3 & 0 \\ 2x^2-1 & x & 0 & x+2 \end{pmatrix}. \tag{3.26}$$

The updated potentials, defined by (3.21) and (3.22), are

$$\rho'' = (-1, 0, -1, -1), \qquad \gamma'' = (3, 6, 4, 4). \tag{3.27}$$

Now, $\widehat{\delta_3}(A''; \rho'', \gamma'') = (0 + (-1) + (-1)) + (6 + 4 + 4) = 12$ and

$$\begin{aligned} I^\sharp &= \{\mathrm{R2}\}, & \tilde{I} &= \{\mathrm{R1}, \mathrm{R2}, \mathrm{R3}, \mathrm{R4}\}, \\ J^\sharp &= \{\mathrm{C2}, \mathrm{C3}, \mathrm{C4}\}, & \tilde{J} &= \{\mathrm{C2}, \mathrm{C3}, \mathrm{C4}\}. \end{aligned}$$

Hence

$$(A'')^\sharp = \begin{array}{cc} & \begin{array}{cccc} & \longleftarrow & J^\sharp & \longrightarrow \end{array} \\ I^\sharp \updownarrow & \begin{pmatrix} 0 & 1 & 0 & 2 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \end{array}.$$

In this case, all of (r1)–(r4) are satisfied. Consequently, we conclude that $\delta_3(A) = \delta_3(A'') = \widehat{\delta_3}(A''; \rho'', \gamma'') = 12$.

Let us explore the other possibility of modifying $A'(x)$. Namely, we change $A'(x)$ according to the modification rule for the case of (r4). Then we have a different modified matrix and a different set of potentials, as follows. We detect the violation of (r4) through the Gaussian row elimination

$$S \cdot (A')^\sharp[I^\sharp, J^\sharp] = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 1 & 1 \\ 0 & 0 \end{pmatrix},$$

in which we note $\rho'_{\mathrm{R2}} > \rho'_{\mathrm{R1}} = \rho'_{\mathrm{R3}}$. Hence we have $H = \{\mathrm{R1}, \mathrm{R3}\}$ and

$$U = \begin{pmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

According to (3.12) we have

$$\begin{aligned} U(x) &= \mathrm{diag}\,(x^{-(-1)}, x^{-0}, x^{-(-1)}, x^{-(-3)}) \cdot U \cdot \mathrm{diag}\,(x^{-1}, x^{0}, x^{-1}, x^{-3}) \\ &= \begin{pmatrix} 1 & -x^{-1} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -x^{-1} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \end{aligned}$$

which modifies $A'(x)$ to

$$A''(x) = U(x)A'(x) = \begin{pmatrix} 0 & -x^{-1} & -x^3 & 2x^3 - x \\ x^5 & x^6 + 1 & x^4 & x^2 \\ x & -x^{-1} & -2x^3 & x \\ 2x^2 & x & 0 & x + 2 \end{pmatrix}.$$

In updating potentials, we have $\beta = 1$ from (3.24) and

$$\rho'' = (-2, 0, -2, -3), \qquad \gamma'' = (5, 6, 5, 5).$$

Note that $A''(x)$ and (ρ'', γ'') defined here are different from those in (3.26) and (3.27). Now, $\widehat{\delta}_3(A''; \rho'', \gamma'') = (0 + (-2) + (-2)) + (6 + 5 + 5) = 12$ and

$$\begin{aligned} I^\sharp &= \{\mathrm{R1}, \mathrm{R2}, \mathrm{R3}\}, \quad \tilde{I} = \{\mathrm{R1}, \mathrm{R2}, \mathrm{R3}\}, \\ J^\sharp &= \{\mathrm{C2}\}, \quad \tilde{J} = \{\mathrm{C1}, \mathrm{C2}, \mathrm{C3}, \mathrm{C4}\}. \end{aligned}$$

Hence

$$(A'')^\sharp = \begin{array}{c|cccc|} & & J^\sharp & & \\ & & \leftrightarrow & & \\ \uparrow & 0 & 0 & -1 & 2 \\ I^\sharp & 1 & 1 & 0 & 0 \\ \downarrow & 0 & 0 & -2 & 0 \\ & 0 & 0 & 0 & 0 \end{array}.$$

In this case, all of (r1)–(r4) are satisfied. Consequently, we conclude again, but using a different $A''(x)$, that $\delta_3(A) = \delta_3(A'') = \widehat{\delta}_3(A''; \rho'', \gamma'') = 12$.

3.6. Algorithm description. We have so far fixed the size k of a submatrix. It is necessary to compute δ_k for $k = 1, 2, \ldots, r$ for obtaining the Smith–McMillan form at infinity.

Before describing the whole algorithm, we should remark how we can detect $\delta_k = -\infty$. For a rational function

$$f(x) = \frac{a_1 x^{d_1} + a_2 x^{d_2} + \cdots + a_\ell x^{d_\ell}}{q(x)} \qquad (d_1 > d_2 > \cdots > d_\ell),$$

we define $\text{min-deg}\, f(x) \equiv d_\ell - \deg q(x)$ with respect to its expression. Put $d_{\min} = \min_{i,j} \text{min-deg}\, A_{ij}(x)$ while $d_{\max} = \max_{i,j} \deg A_{ij}(x)$. If δ_k is finite, it must satisfy $\delta_k \geq k \cdot d_{\min}$. Hence we can conclude that $\delta_k = -\infty$ when $\widehat{\delta}_k(A; \rho, \gamma) < k \cdot d_{\min}$ for some admissible (ρ, γ).

We now summarize the algorithm for computing δ_k for $k = 1, 2, \ldots, r (= \mathrm{rank}\, A)$ as follows. It is not necessary to input r, since it is computed in the procedure.

ALGORITHM FOR COMPUTING $\delta_k(A)$ $(k = 1, 2, \ldots)$.

Step 0: Compute $d_{\min}$. Set $k := 1$. Find a set of initial potentials (ρ, γ) which satisfies $\rho_i + \gamma_j \geq c_{ij}$ for $i \in R$ and $j \in C$.

Step 1: Compute

$$\widehat{\delta_k} := \sum_{i=1}^{k} \rho_{\tau(i)} + \sum_{j=1}^{k} \gamma_{\sigma(j)},$$

where τ and σ are permutations such that $\rho_{\tau(1)} \geq \rho_{\tau(2)} \geq \cdots \geq \rho_{\tau(m)}$ and $\gamma_{\sigma(1)} \geq \gamma_{\sigma(2)} \geq \cdots \geq \gamma_{\sigma(n)}$. If $\widehat{\delta_k} < k \cdot d_{\min}$, then halt (at this point $(k-1)$ equals the rank r).

Step 2: Put

$$I^\sharp := \{i \in R \mid \rho_i > \rho_{\tau(k+1)}\}, \qquad \tilde{I} := \{i \in R \mid \rho_i \geq \rho_{\tau(k)}\},$$
$$J^\sharp := \{j \in C \mid \gamma_j > \gamma_{\sigma(k+1)}\}, \qquad \tilde{J} := \{j \in C \mid \gamma_j \geq \gamma_{\sigma(k)}\};$$

where exceptionally $I^\sharp = R$ and $J^\sharp = C$ in case $k = m$ and $k = n$, respectively. Put

$$A^\sharp_{ij} := \begin{cases} \lim_{x\to\infty} x^{-\rho_i-\gamma_j} A_{ij}(x) & \text{if } i \in \tilde{I} \text{ and } j \in \tilde{J}, \\ 0 & \text{otherwise.} \end{cases}$$

If the four conditions (r1)–(r4) in Proposition 3.1 are satisfied, then output $\widehat{\delta_k}$. If at least one condition of the four is violated, go to Step 3. If $k = \min(m, n)$, then halt. Otherwise $k := k + 1$ and go to Step 1.

Step 3: Modify the matrix $A(x)$ and the potentials as described in (3.17)–(3.23), according to which of (r1)–(r4) is violated. Go to Step 1.

In the algorithm described above, we may start with an arbitrary set of admissible potentials. It is desirable, however, to adopt reasonably small integers as the initial potentials.

One of the natural choices is to put

$$\rho_i = \max_{j\in C} c_{ij} \quad \text{for } i \in R, \qquad \gamma_j = 0 \quad \text{for } j \in C. \tag{3.28}$$

In this case, $\gamma_j = 0$ for $j \in C$ throughout the computation. Hence ρ_i for $i \in R$ always coincide with the row degrees. Since $J^\sharp = \emptyset$ and $\tilde{J} = C$, the four rank conditions in the test for tightness (see Proposition 3.1) are reduced to a couple of conditions, (r1) and (r2). This algorithm will be called the row-degree method, and is an extension of the well-known algorithm for $\delta_m(A)$ of an $m \times n$ row-full rank matrix $A(x)$ based on the concept of row properness (Wolovich [24]).

Another natural choice is, of course, the symmetric (transposed) version of the above, i.e.,

$$\rho_i = 0 \quad \text{for } i \in R, \qquad \gamma_j = \max_{i\in R} c_{ij} \quad \text{for } j \in C. \tag{3.29}$$

This algorithm, however, does not behave in a manner symmetric to the row-degree method. This is because the symmetry between rows and columns is broken in the process of modification. In fact, any condition of the four may possibly be violated. Thus this algorithm is not the "column-degree method," and the primal-dual framework is not a simple extension of the Wolovich algorithm.

3.7. Complexity. In this section, we analyze the time complexity of our algorithm. First we consider the number of possible modifications when k ranges from 2 to r.

LEMMA 3.4. *The total number of modifications for $k = 2, \ldots, r$ is bounded by $r(d_{\max} - d_{\min})$.*

Proof. Let $A^{(\ell)}$ denote the matrix constructed for $k = \ell$ such that $\delta_\ell(A) = \delta_\ell(A^{(\ell)}) = \widehat{\delta}_\ell(A^{(\ell)})$, where the potentials are omitted for notational simplicity. First note $\delta_{\ell\pm1}(A^{(\ell)}) \leq \widehat{\delta}_{\ell\pm1}(A^{(\ell)})$ and $\widehat{\delta}_{\ell+1}(A^{(\ell)}) \leq 2\widehat{\delta}_\ell(A^{(\ell)}) - \widehat{\delta}_{\ell-1}(A^{(\ell)})$ (concavity). Then (the number of modifications for $k = \ell + 1$) $\leq \widehat{\delta}_{\ell+1}(A^{(\ell)}) - \delta_{\ell+1}(A^{(\ell)}) \leq 2\delta_\ell(A) - \delta_{\ell-1}(A) - \delta_{\ell+1}(A)$. Therefore, the total number of modifications in the course of computation for $k = 2, \ldots, r$ is bounded as follows:

$$\begin{aligned}\sum_{\ell=1}^{r-1}(\widehat{\delta}_{\ell+1}(A^{(\ell)}) - \delta_{\ell+1}(A^{(\ell)})) &\leq \delta_1(A) + \delta_{r-1}(A) - \delta_r(A)\\ &\leq d_{\max} + (r-1)\cdot d_{\max} - r\cdot d_{\min}\\ &= r\cdot(d_{\max} - d_{\min}). \qquad \square\end{aligned}$$

Consequently, from Lemma 3.4, the number of modifications is $O(r)$ provided that $(d_{\max} - d_{\min})$ is a constant, which is the case with matrix pencils.

Next, we consider the time complexity with respect to m and n (the size of the matrix) and r (the rank of A).

THEOREM 3.5. *The time complexity of the proposed algorithm for computing all the δ_k is $O(rmn\max(m, n))$, provided that $(d_{\max} - d_{\min})$ is a constant.*

Proof. The time complexity for computing δ_k for $k = 1, 2, \ldots, r$ can be estimated by (r+the number of modifications)$\times O(mn\max(m, n))$. According to Lemma 3.4, the number of modifications is $O(r)$, and therefore the whole time complexity is $O(rmn\max(m, n))$. $\square$

3.8. Extension. We will extend the proposed algorithm so as to compute

$$\delta_k(A; I_0, J_0) = \max_{I,J}\{\deg\det A[I, J] \mid I \supseteq I_0, J \supseteq J_0, |I| = |J| = k\},$$

the maximum degree of a minor of order k, which covers the specified row-set I_0 and column-set J_0.

For (ρ, γ) admissible in the sense of (3.2) we define

$$\widehat{\delta}_k(A; I_0, J_0; \rho, \gamma) = \max_{I \supseteq I_0, |I|=k} \rho(I) + \max_{J \supseteq J_0, |J|=k} \gamma(J),$$

and then we have

$$\delta_k(A; I_0, J_0) \leq \widehat{\delta}_k(A; I_0, J_0; \rho, \gamma). \tag{3.30}$$

With permutations τ and σ such that

$$\tau : \begin{cases} \{1, \ldots, |I_0|\} \to I_0, & \\ \{|I_0| + 1, \ldots, m\} \to R - I_0 & \text{such that } \rho_{\tau(|I_0|+1)} \geq \rho_{\tau(|I_0|+2)} \geq \cdots \geq \rho_{\tau(m)}; \end{cases}$$

$$\sigma : \begin{cases} \{1, \ldots, |J_0|\} \to J_0, & \\ \{|J_0| + 1, \ldots, n\} \to C - J_0 & \text{such that } \gamma_{\sigma(|J_0|+1)} \geq \gamma_{\sigma(|J_0|+2)} \geq \cdots \geq \gamma_{\sigma(n)}, \end{cases}$$

it holds that

$$\widehat{\delta}_k(A; I_0, J_0; \rho, \gamma) = \sum_{i=1}^{k} \rho_{\tau(i)} + \sum_{j=1}^{k} \gamma_{\sigma(j)}.$$

We redefine $I^\sharp$, $J^\sharp$, $\tilde{I}$, and $\tilde{J}$ as follows:

$$I^\sharp = \{i \in (R - I_0) \mid \rho_i > \rho_{\tau(k+1)}\}, \qquad \tilde{I} = \{i \in (R - I_0) \mid \rho_i \geq \rho_{\tau(k)}\},$$
$$J^\sharp = \{j \in (C - J_0) \mid \gamma_j > \gamma_{\sigma(k+1)}\}, \qquad \tilde{J} = \{j \in (C - J_0) \mid \gamma_j \geq \gamma_{\sigma(k)}\}.$$

Exceptionally, in case $k = m$, we set $I^\sharp = \tilde{I} = R - I_0$, and in case $k = n$, $J^\sharp = \tilde{J} = C - J_0$.

The definition of the constant matrix $A^\sharp$ is also modified:

$$A^\sharp_{ij} = \begin{cases} \lim\limits_{x\to\infty} x^{-\rho_i-\gamma_j} A_{ij}(x) & \text{if } i \in (I_0 \cup \tilde{I}) \text{ and } j \in (J_0 \cup \tilde{J}), \\ 0 & \text{otherwise.} \end{cases}$$

Proposition 3.1 for tightness is rewritten as follows.

PROPOSITION 3.6 (tightness). *The following three conditions are equivalent.*

(a) $\widehat{\delta}_k(A; I_0, J_0) = \delta_k(A; I_0, J_0)$.

(b) *There exist* $I \supseteq I_0 \cup I^\sharp$ *and* $J \supseteq J_0 \cup J^\sharp$ *such that* $\operatorname{rank} A^\sharp[I, J] = |I| = |J| = k$.

(c) *The following four conditions are satisfied:*

(r1) $\operatorname{rank} A^\sharp[R, C] \geq k$;

(r2) $\operatorname{rank} A^\sharp[I_0 \cup I^\sharp, C] = |I_0 \cup I^\sharp|$;

(r3) $\operatorname{rank} A^\sharp[R, J_0 \cup J^\sharp] = |J_0 \cup J^\sharp|$;

(r4) $\operatorname{rank} A^\sharp[I_0 \cup I^\sharp, J_0 \cup J^\sharp] \geq |I_0 \cup I^\sharp| + |J_0 \cup J^\sharp| - k$.

The matrix $A(x)$ is modified on the basis of the following fact. For simplicity, I_0 and J_0 are assumed to be the first $|I_0|$ rows and the first $|J_0|$ columns. Then $A'(x) = U(x)A(x)V(x)$ satisfies $\delta_k(A') = \delta_k(A)$ if

$$U(x) = \begin{array}{c|c|c} & I_0 & R - I_0 \\ \hline I_0 & U_0(x) & O \\ R - I_0 & U_2(x) & U_1(x) \end{array}, \qquad V(x) = \begin{array}{c|c|c} & J_0 & C - J_0 \\ \hline J_0 & V_0(x) & V_2(x) \\ C - J_0 & O & V_1(x) \end{array},$$

where $U_0(x)$ is an $|I_0| \times |I_0|$ matrix with $\deg\det U_0(x) = 0$, $U_1(x)$ is a biproper matrix, and $U_2(x)$ is an arbitrary rational matrix; and $V_0(x)$ is a $|J_0| \times |J_0|$ matrix with $\deg\det V_0(x) = 0$, $V_1(x)$ is a biproper matrix, and $V_2(x)$ is an arbitrary rational matrix.

4. Experimental evaluation.

4.1. Method. We perform experiments on Sun SPARC Station 10 to evaluate the three combinatorial relaxation-type algorithms, the primal-dual method (PD) with initial potentials given by (3.29), the row-degree method (RD), and the matching method (M). All of these are implemented in C language. For comparison, we also realize in Maple (a computer algebraic system) the elimination method (E) described in §2.

Our program restricts input matrices to Laurent polynomial matrices of which each entry is a polynomial in x and $1/x$.

We provide two types of sample problems, regular pencils (P) and band matrices (B). A regular pencil $xE - F$ used here is generated randomly in the following way:

$$E_{ij} = \begin{cases} g & \text{with probability } \frac{1-p}{2K} \quad \text{for } g = \pm1, \pm2, \ldots, \pm K, \\ 0 & \text{with probability } p; \end{cases}$$

$$F_{ij} = \begin{cases} h & \text{with probability } \frac{1-q}{2K} \quad \text{for } h = \pm1, \pm2, \ldots, \pm K, \\ 0 & \text{with probability } q. \end{cases}$$

We put $K = 1$, $p = 0.0625$, and $q = 0.5$.

On the other hand, we provide band matrices in the following way. First we put

$$X = \operatorname{diag}(x; \mathbf{d}), \quad \mathbf{d} = (d_1, d_2, \ldots d_n),$$

where d_i is generated randomly as follows:

$$d_i = \ell \quad \text{with probability } \frac{1}{2L+1} \quad \text{for } \ell = 0, \pm 1, \pm 2, \ldots, \pm L.$$

With a lower bidiagonal matrix

$$P = \begin{pmatrix} 1 & & & \\ \frac{1}{x} & 1 & & \\ & \ddots & \ddots & \\ & & \frac{1}{x} & 1 \end{pmatrix}$$

and an upper bidiagonal matrix

$$Q = \begin{pmatrix} 1 & \frac{1}{x} & & \\ & \ddots & \ddots & \\ & & 1 & \frac{1}{x} \\ & & & 1 \end{pmatrix}$$

we construct a pentadiagonal (band) matrix $A(x) = P^2XQ^2$. We put $L = 10$. Obviously, both P and Q are biproper, and hence the Smith–McMillan form at infinity of $A(x)$ is the same as that of X, which is easily obtained by sorting $(d_1, \ldots, d_n)$. Therefore, we know $\delta_k(A)$ in advance, and we use this type of matrix as sample data to ensure that our program returns correct answers.

We measure CPU time and the number of modifications (for PD, RD, and M) to evaluate each algorithm.

4.2. Results. Before we report experimental results, we should recall that the time complexity is $O(rmn \max(m, n))$ if $(d_{\max} - d_{\min})$ is bounded by a constant (see Theorem 3.5). In fact, we have $(d_{\max} - d_{\min}) \leq 1$ with the pencils and $(d_{\max} - d_{\min}) \leq 2L + 2$ with the band matrices. Therefore, the time complexity is $O(rmn \max(m, n))$.

Table 1 and Figure 1 show the results for the pencils, and Table 2 and Figure 2 show the results for the band matrices. In Figures 1 and 2, the abscissa is $\log_2 n$, where n is the size of a matrix, while the ordinate is $\log_{10} T$, where T is the CPU time. Algorithm PD failed for $n = 128$ because of a memory shortage. The experiments with algorithm E for large n were given up because it would have taken many days.

It is observed in our experiment that for an $n \times n$ matrix A, approximately $O(n^{3.4})$ time is needed by each combinatorial relaxation algorithm, while time complexity given by Theorem 3.5 is $O(n^4)$ since rank $A = n$ in both problem types P and B. On the other hand, the elimination method takes approximately $O(n^4)$ time in our experiment, whereas the number of operations on rational functions is $O(n^3)$, as mentioned in §2.

In the above experiment, no distinction in computing time is recognized among algorithms PD, RD, and M. Since algorithm RD lacks symmetry, it is natural that this algorithm takes longer than the other algorithms for a matrix with a special structure. For example, let us consider

$$A(x) = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & x & x^2 & \cdots & x^{n-1} \\ 1 & x^2 & x^4 & \cdots & x^{2(n-1)} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & x^{n-1} & x^{2(n-1)} & \cdots & x^{(n-1)^2} \end{pmatrix}. \tag{4.1}$$

The results for the matrix with $n = 16$ are shown in Table 3.

TABLE 1

Statistics over 10 *pencils of each size. CPU time in seconds and the number of modifications show the mean values of* 10 *iterations* ($K = 1$, $p = 0.0625$, $q = 0.5$; *Sun SPARC Station* 10).

Algorithms		n				
		8	16	32	64	128
PD	CPU time (s) Stand. Dev. # Modifications	6.67×10^{-3} 8.61×10^{-3} 0.3	6.17×10^{-2} 8.05×10^{-3} 0.9	6.72×10^{-1} 1.77×10^{-2} 1.0	7.73 1.20×10^{-1} 0.6	— — —
RD	CPU time (s) Stand. Dev. # Modifications	1.17×10^{-2} 8.05×10^{-3} 0.7	5.67×10^{-2} 8.61×10^{-3} 1.1	5.70×10^{-1} 2.19×10^{-2} 0.8	6.91 1.04×10^{-1} 0.7	9.87×10 5.33×10^{-1} 0.1
M	CPU time (s) Stand. Dev. # Modifications	1.17×10^{-2} 8.05×10^{-3} 0.1	6.83×10^{-2} 9.46×10^{-3} 0.3	6.62×10^{-1} 3.04×10^{-2} 0.2	7.61 1.41×10^{-1} 0.1	1.04×10^{2} 4.56×10^{-1} 0.0
E	CPU time (s) Stand. Dev.	3.23 4.83×10^{-1}	4.41×10 4.94	1.10×10^{3} 2.11×10	— —	— —

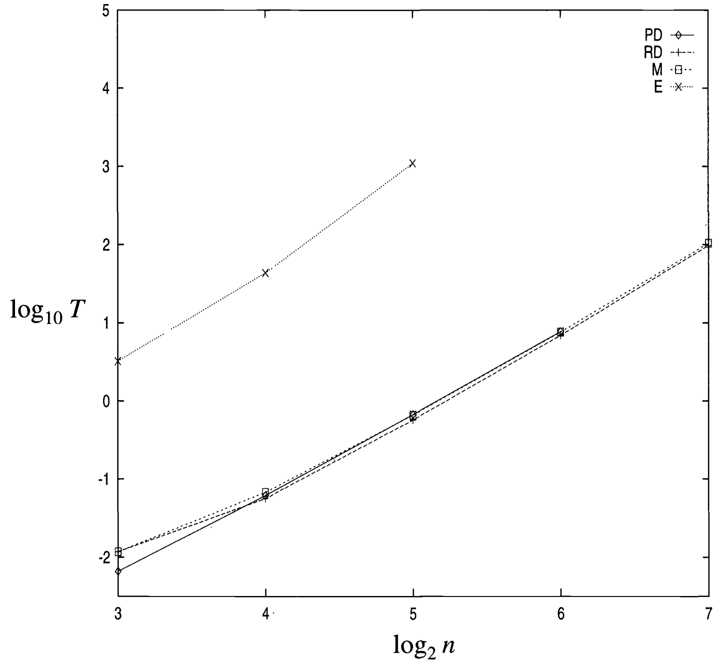


FIG. 1. *Results for pencils. The abscissa is* $\log_2 n$, *where n is the size of a matrix, while the ordinate is* $\log_{10} T$, *where T denotes CPU time* ($K = 1$, $p = 0.0625$, $q = 0.5$; *Sun SPARC Station* 10).

5. Discussion. First we compare various aspects of the three combinatorial relaxation algorithms: accuracy, running time, and facility in implementation.

Numerical difficulty is inherent in the problem of computing the maximum degree of subdeterminants, and hence inevitable with any kind of algorithm. We intend, however, to exert the greatest possible effort to circumvent this numerical difficulty by making use of combinatorial (or structural) information. The total amount of numerical computation, which causes the accumulation of rounding errors, increases in proportion to the number of modifications in combinatorial relaxation algorithms. In this regard we recommend method M among the combinatorial relaxation algorithms, because it usually requires fewer modifications.

TABLE 2
Statistics over 10 *band matrices of each size. CPU time in seconds and the number of modifications show the mean values of* 10 *iterations* ($L = 10$; *Sun SPARC Station* 10).

Algorithms		n				
		8	16	32	64	128
PD	CPU time (s)	1.50×10^{-2}	1.03×10^{-1}	9.92×10^{-1}	1.19×10	—
	Stand. Dev.	5.27×10^{-3}	1.31×10^{-2}	3.54×10^{-2}	3.22×10^{-1}	—
	# Modifications	6.5	12.1	15.9	18.1	—
RD	CPU time (s)	1.50×10^{-2}	9.83×10^{-2}	1.01	1.11×10	1.41×10^{2}
	Stand. Dev.	5.27×10^{-3}	1.46×10^{-2}	4.92×10^{-2}	2.03×10^{-1}	7.38×10^{-1}
	# Modifications	5.5	11.9	15.7	18.1	18.9
M	CPU time (s)	1.50×10^{-2}	1.13×10^{-1}	9.37×10^{-1}	1.04×10	1.35×10^{2}
	Stand. Dev.	1.23×10^{-2}	1.89×10^{-2}	2.81×10^{-2}	2.57×10^{-1}	1.81
	# Modifications	3.6	6.8	8.5	10.2	11.6
E	CPU time (s)	2.92	4.52×10	7.65×10^{2}	1.28×10^{4}	—
	Stand. Dev.	1.12	4.13×10	6.94×10^{2}	1.27×10^{4}	—

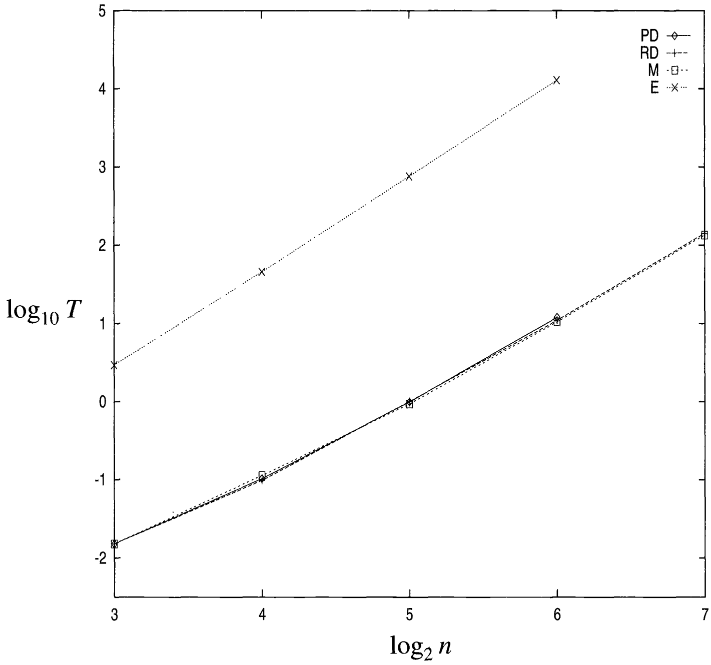


FIG. 2. *Results for band matrices. The abscissa is* $\log_2 n$, *where n is the size of a matrix, while the ordinate is* $\log_{10} T$, *where T denotes CPU time* ($L = 10$; *Sun SPARC Station* 10).

TABLE 3
CPU time in seconds and the number of modifications for the matrix $A(x)$ *of* (4.1) *with* $n = 16$.

Algorithms	CPU time (s)	# Modifications
PD	1.67	139
RD	2.03	156
M	8.33×10^{-2}	0

When $r = \text{rank}\, A(x) < \min(m, n)$, however, even the recommended method, M, requires a large number of modifications and hence suffers from numerical difficulty in detecting $\delta_k(A) = -\infty$ for $k = r + 1$. In fact, we often encountered this numerical difficulty in experiments with some singular pencils. If we know the rank r in advance, we should therefore stop the computation at $k = r$.

As to running time, the experimental results show that although we take more time to compute the estimate in method M than in method PD or RD, the whole running time of these three are almost equal. This is due to the fact that M involves a smaller number of modifications.

Let us focus on facility in implementation. In this respect, method PD, as well as method RD, is advantageous, because it does not rely on any sophisticated algorithms in combinatorial optimization, whereas method M depends on a refined algorithm for the weighted matching problem.

Next we discuss the applicability of our programs of combinatorial relaxation algorithms. Recall that our implementation is restricted to Laurent polynomial matrices. In order to make our program capable of dealing with rational function matrices, we have to use simplification procedures for rational functions, which are realized in computer algebraic systems. Taking this into consideration, the combinatorial relaxation-type algorithm for the maximum degree of subdeterminants is expected to be implemented in computer algebraic systems.

6. Conclusion. In this paper, we have proposed a primal-dual framework of combinatorial relaxation algorithms for computing the maximum degree of subdeterminants. Furthermore, we have realized combinatorial relaxation algorithms in computer programs and made experiments. It turns out that combinatorial relaxation algorithms are effective for computing the maximum degree of subdeterminants and the index of nilpotency.

The numerical difficulty is inherent in the problem. In particular, the index of nilpotency is very sensitive to perturbations unless it coincides with that obtained under a genericity assumption. Hence the structural approach has been taken extensively. When we model practical engineering systems such as electric networks or chemical plants, however, we often encounter numerical cancellations (or degeneracy) arising from fixed constants or an algebraic relationship among system parameters. In this case the naive genericity assumption is not valid, and the answer of a naive structural approach loses its physical significance. Hence it is necessary to use more than the naive structural approaches. We may employ a more sophisticated framework in the structural approach, as in [13, Chap. 4] and [17], on the basis of the observation that this kind of cancellation phenomena is robust against perturbations to physical parameters, because such constants or relations are determined by the combinatorial or geometric structure of the system rather than by physical characteristics. Alternatively, we may resort to algorithms of the kind treated in the present paper. We conclude this paper by mentioning that our algorithms gave the correct answer $\nu = 2$ to the example problem from [22] described in the Introduction.

Acknowledgments. The authors are grateful to Dr. Pawel Bujakiewicz for providing a number of relevant references, to Professor Kokichi Sugihara for his support, and to Dr. Keiichi Kaneko for advice on implementation. They also appreciate essential comments by anonymous referees that improved the presentation.

REFERENCES

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows — Theory, Algorithms and Applications*, Prentice Hall, Englewood Cliffs, NJ, 1993.

[2] K. E. Brenan, S. L. Campbell, and L. R. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, North-Holland, Amsterdam, 1989.

[3] R. A. Brualdi and H. J. Ryser, *Combinatorial Matrix Theory*, Cambridge University Press, UK, 1991.

[4] P. Bujakiewicz, *Maximum Weighted Matching for High Index Differential Algebraic Equations*, Ph.D. thesis, Delft University of Technology, the Netherlands, 1994.

[5] C. Commault and J.-M. Dion, *Structure at infinity of linear multivariable systems: A geometric approach*, IEEE Trans. Automat. Control, AC-27 (1982), pp. 693–696.

[6] C. Commault, J.-M. Dion, and A. Perez, *Disturbance rejection for structured systems*, IEEE Trans. Automat. Control, AC-36 (1991), pp. 884–887.

[7] I. Duff and C. W. Gear, *Computing the structural index*, SIAM J. Algebraic Discrete Meth., 7 (1986), pp. 594–603.

[8] R. Gani and I. T. Cameron, *Modelling for dynamic simulation of chemical processes: The index problem*, Chem. Engrg. Sci., 47 (1992), pp. 1311–1315.

[9] C. W. Gear, *Differential-algebraic equation index transformations*, SIAM J. Sci. Stat. Comput., 9 (1988), pp. 39–47.

[10] ———, *Differential algebraic equations, indices, and integral algebraic equations*, SIAM J. Numer. Anal., 27 (1990), pp. 1527–1534.

[11] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations* II, Springer-Verlag, Berlin, 1991.

[12] M. L. J. Hautus, *The formal Laplace transform for smooth linear systems*, in Mathematical Systems Theory, G. Marchesini and S. K. Mitter, eds., Lecture Notes in Economics and Mathematical Systems 131, Springer-Verlag, Berlin, 1976, pp. 29–47.

[13] K. Murota, *Systems Analysis by Graphs and Matroids — Structural Solvability and Controllability*, Algorithms and Combinatorics 3, Springer-Verlag, Berlin, 1987.

[14] ———, *Computing Puiseux-series solutions to determinantal equations via combinatorial relaxation*, SIAM J. Comput., 19 (1990), pp. 1132–1161.

[15] ———, *Combinatorial relaxation algorithm for the maximum degree of subdeterminants: Computing Smith–McMillan form at infinity and structural indices in Kronecker form*, Appl. Algebra Engrg. Comm. Comput., 6 (1995), pp. 251–273.

[16] ———, *Computing the degree of determinants via combinatorial relaxation*, SIAM J. Comput., 24 (1995), pp. 765–796.

[17] K. Murota and J. W. van der Woude, *Structure at infinity of structured descriptor systems and its applications*, SIAM J. Control Optim., 29 (1991), pp. 878–894.

[18] M. Newman, *Integral Matrices*, Academic Press, New York, 1972.

[19] C. C. Pantelides, *The consistent initialization of differential-algebraic systems*, SIAM J. Sci. Stat. Comput., 9 (1988), pp. 213–231.

[20] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice Hall, Englewood Cliffs, NJ, 1982.

[21] N. Suda, B. Wan, and I. Ueno, *The orders of infinite zeros of structured systems*, Trans. Soc. Instr. Contr. Engrg. Japan, 25 (1989), pp. 1062–1068. (In Japanese.)

[22] J. Ungar, A. Kröner, and W. Marquardt, *Structural analysis of differential-algebraic equation systems — Theory and application*, Comput. Chem. Engrg., 19 (1995), pp. 867–882.

[23] G. C. Verghese and T. Kailath, *Rational matrix structure*, IEEE Trans. Automat. Control, AC-26 (1981), pp. 434–439.

[24] W. A. Wolovich, *Linear Multivariable Systems*, Springer-Verlag, Berlin, 1974.

[25] J. W. van der Woude, *On the structure at infinity of a structured system*, Linear Algebra Appl., 148 (1991), pp. 145–169.

SIAM J. SCI. COMPUT.
Vol. 17, No. 4, pp. 1013–1017, July 1996

014

TIMELY COMMUNICATION

Under the "timely communications" policy for the SIAM Journal on Scientific Computing, papers that have significant timely content and do not exceed five pages automatically will be considered for a separate section of the journal with an accelerated reviewing process. It will be possible for the note to appear approximately six months after the date of acceptance.

VERIFICATION MAY BE BETTER THAN ESTIMATION*

C. FALCÓ KORN[†], B. HÖRMANN[‡], AND C. P. ULLRICH[§]

Abstract. Error bounds for an approximate solution of a linear system of equations are usually derived from backward error analysis. It is known that such error bounds do not guarantee a certain accuracy in the approximation. On the other hand, verification provides componentwise guaranteed bounds. A series of examples gives evidence that for H-matrices error verification may be better than traditional error estimation with respect to both the quality of the delivered bounds and computational performance.

Key words. error analysis, error estimation, verification, linear systems, numerical libraries, LAPACK, ITPACK

AMS subject classifications. 65G05, 65G10

1. Introduction.

Wilkinson showed that a computer solution is the exact solution of a nearby problem thus initiating backward error analysis. Subsequently, main results from this theory were incorporated into all major numerical packages like ITPACK, LINPACK, or LAPACK. But backward error analysis is not able to guarantee lower bounds on the number of correct digits in the approximation. In this paper we outline the basic idea behind error verification and demonstrate its effectiveness by a series of examples.

2. Backward error estimation.

Let $\widehat{x}$ be an approximation of the solution to the linear system $Ax = b$. The (normwise) backward error is defined by

$$\min_{E,f}\left(\max\left(\frac{\|E\|}{\|A\|},\frac{\|f\|}{\|b\|}\right)\right)$$

such that $\widehat{x}$ satisfies $(A+E)\widehat{x} = b+f$. The (normwise) forward error $\|x-\widehat{x}\|/\|x\|$ is related to the backward error by the condition number $\kappa(A) = \|A\| \cdot \|A^{-1}\|$. Indeed, if $\|E\| \leq \delta\|A\|$ and $\|f\| \leq \delta\|b\|$, Theorem 2.7.2 in [7] gives the bound

$$\frac{\|x-\widehat{x}\|}{\|x\|} \leq \frac{2\delta}{1-r}\kappa(A),$$

where $r = \delta\kappa(A)$ must be less then 1.

Although all packages compute the error (and thus the accuracy) in terms of the underlying backward error, definitions for the backward error and the condition number vary. (See, for example, [1], where a condition number is proposed which depends not only on the data matrix but on the approximation $\widehat{x}$ and the right-hand side b as well.) In practice, however, the exact condition number is not known. Moreover, the normwise forward error cannot be translated into a statement about the number of correct digits if the components in $\widehat{x}$ are not of the same order of magnitude. (See [7, Ex. 2.2.1, p. 54].)

*Received by the editors September 21, 1994; accepted for publication (in revised form) January 19, 1996.

[†] The London Parallel Applications Centre, Queen Mary & Westfield College, Mile End Road, London E1 4NS, England (C.Falco-Korn@lpac.ac.uk).

[‡] Institut für Informatik, Universität Basel, Mittlere Strasse 142, 4056 Basel, Switzerland (hoermann@ifi.unibas.ch).

[§] Institut für Informatik, Universität Basel, Mittlere Strasse 142, 4056 Basel, Switzerland (ullrich@ifi.unibas.ch).

3. Verified error bounds. In contrast to an error estimating routine, an error verifying routine determines upper bounds for the vector of relative errors which is defined as

$$t_i = \frac{|x_i - \widehat{x}_i|}{|x_i|} \quad (x_i \neq 0),$$

where again $\widehat{x}$ is an approximation and x is the exact solution of the linear system. Thus, a vector w from a verifying routine satisfies $w_i \geq t_i$. It tells us that at least $\lfloor -\log(w_i) \rfloor$ digits are correct in the ith component of the approximation. Furthermore, all components in the approximation are correct to at least $\min_i(\lfloor -\log(w_i)\rfloor)$ digits. We say that $\widehat{x}$ has a *true accuracy* of $\min_i(\lfloor -\log(t_i)\rfloor)$ digits. We call $\min_i(\lfloor -\log(w_i)\rfloor)$ the *lowest verified accuracy* and $\max_i(\lfloor -\log(w_i)\rfloor)$ the *highest verified accuracy*.

Error verifying routines were implemented by Falcó Korn [2]. His thesis provides the whole theoretical background. A concise English description can be found in [3, 4].

4. Examples. We preset the solution x and compute the right-hand side $b = Ax$ by a matrix–vector product. The right-hand sides will be $b1 = Ax1$ where all components of $x1$ are equal to 1, $b2 = Ax2$ with the ith component of $x2$ set to $1/i$, and $b3 = Ax3$ with $x3$ like $x1$ but with the last component set to 2^{50}. Examples 1 and 2 are computed on a NeXT workstation (NeXTstation II) and Example 3 is computed on a Macintosh IIci. Floating-point numbers conform to the IEEE double format which corresponds to at least 16 decimal digits.

4.1. Example 1. LAPACK [1] provides two different error estimation methods, both derived from backward error analysis. The first one (*Estimation* 1) is based on the DxxCON routine (where xx denotes one of LAPACK's storage schemes) which estimates the normwise condition number (see §2). The second one (*Estimation* 2) is computed with the DxxRFS routine. It calculates a componentwise backward error and an estimation of a componentwise condition number depending on A, b, and the approximation $\widehat{x}$.

For large narrow banded systems, Estimation 1 is feasible only for symmetric, positive definite, tridiagonal matrices [9]. To compare results over a wide range of dimensions, we consider the one-dimensional discrete Laplacian

$$(1) \qquad A = \underbrace{\begin{bmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix}}_{n}.$$

First we compute approximations of $Ax = bi$ $(i = 1, 2, 3)$ using LAPACK's DPTTRF and DPTTRS routines [1]. Then we estimate the error using Estimations 1 and 2 for xx = PT and compare their results to the output of a verifying routine; see Table 1. Note that figures in the rows labeled with *Estimation* 1 and *Estimation* 2 denote $\lfloor -\log(\epsilon)\rfloor$, where ϵ denotes the normwise forward error as given by the corresponding LAPACK routines.

To summarize, accuracy statements from either estimation or verification are comparable to each other as long as all components of the solution vector are of the same order of magnitude. If this is not the case, overestimations may occur (see the last five rows). This is a consequence of the remarks at the end of §2.

For this example, verification is significantly slower than both estimations. Since no PT verifying routine is available yet, such a case is mapped onto a verifying routine which operates on PB-matrices.

TABLE 1

True accuracy, estimated accuracies, and verified accuracies of three approximations corresponding to the linear systems with matrix (1) *and right-hand sides* b1, b2, *and* b3. *LAPACK's PT routines were used for Estimations* 1 *and* 2. *Figures in the parentheses indicate execution times normalized by the time needed to compute Estimation* 1 *for a dimension of* 10^6.

	Dimension	10^2	10^3	10^4	10^5	10^6	
	True accuracy	13	12	10	9	6	
	Estimation 1	12	10	8	6	4	(1)
b1	Estimation 2	11	9	7	5	3	(2)
	Lowest verified	11	9	7	5	3	} (4)
	Highest verified	13	12	11	10	9	
	True accuracy	13	12	10	9	6	
	Estimation 1	12	10	8	6	4	(1)
b2	Estimation 2	11	9	7	5	3	(4)
	Lowest verified	11	9	7	5	3	} (4)
	Highest verified	14	14	13	13	13	
	True accuracy	6	12	6	6	6	
	Estimation 1	12	10	8	6	4	(1)
b3	Estimation 2	11	9	7	5	3	(2)
	Lowest verified	5	5	5	5	3	} (4)
	Highest verified	14	14	14	14	14	

Now consider the two-dimensional discrete Laplacian:

$$(2) \qquad A = \underbrace{\begin{bmatrix} B & C & & \\ C & \ddots & \ddots & \\ & \ddots & \ddots & C \\ & & C & B \end{bmatrix}}_{n}, B = \underbrace{\begin{bmatrix} 4 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 4 \end{bmatrix}}_{\sqrt{n}}, C = \underbrace{\begin{bmatrix} -1 & & \\ & \ddots & \\ & & -1 \end{bmatrix}}_{\sqrt{n}}.$$

Since this matrix is not tridiagonal, we use LAPACK's PB routines. Table 2 summarizes the results. The verifying routine is now significantly faster than both estimations. For the case where the right-hand side is $b3$, it gives the only realistic answer.

4.2. Example 2. For dense matrices the dominant time factor is the factorization of the matrix. In cases where L and U are computed once to solve several linear systems, we compare the time for the computation of an approximation $\widetilde{x}$ with the time needed to obtain a statement about its accuracy.

The following strictly diagonally dominant matrix is used in [8], a statistical study on condition numbers and estimation methods:

$$(3) \qquad A = (a_{ij}) = \begin{cases} \frac{i-j}{i+j-1}, & i \neq j, \\ n, & i = j. \end{cases}$$

We proceed as in Example 1. Since the matrix is dense we use LAPACK's GE routines. Table 3 lists execution times normalized by the time needed to compute $\widetilde{x}$. It shows a typical situation for dense systems: Estimation 1 is faster than Estimation 2, while the verifying routine offers the fastest way to obtain an error bound.

The accuracy results for this example are listed in Table 4. LAPACK's Estimation 1 overestimates the true accuracy.

4.3. Example 3. Iterative solvers stop when the backward error reaches some prescribed threshold. (See Chapter 4.2 of [10].) Since ITPACK [6] does not provide condition estimators,

TABLE 2

True accuracy, estimated accuracies, and verified accuracies of three approximations corresponding to the linear systems with matrix (2) *and right-hand sides* b1, b2, *and* b3. *LAPACK's PB routines were used for Estimations* 1 *and* 2. *Figures in the parentheses indicate execution times normalized by the time needed to compute Estimation* 1 *for a dimension of* 6400.

	Dimension	1024	2500	6400	
	True accuracy	14	13	13	
	Estimation 1	13	12	12	(1)
b1	Estimation 2	11	10	10	(0.2)
	Lowest verified	11	11	10	} (0.04)
	Highest verified	13	13	13	
	True accuracy	14	13	13	
	Estimation 1	13	12	12	(1)
b2	Estimation 2	13	13	13	(0.2)
	Lowest verified	11	11	10	} (0.04)
	Highest verified	14	14	14	
	True accuracy	6	7	6	
	Estimation 1	13	12	12	(1)
b3	Estimation 2	13	13	13	(0.2)
	Lowest verified	5	5	5	} (0.04)
	Highest verified	14	14	14	

TABLE 3

Runtime ratios solving a linear system with matrix (3) *and right-hand side* b1. *LAPACK's GE routines were used for the computation of $\hat{x}$ and for both estimations.*

Dimension	300	600	900	1200
Computation of $\hat{x}$	1	1	1	1
Estimation 1	4.7	4.5	4.5	4.5
Estimation 2	12	14	14	14
Verification	1.7	1.7	1.7	1.8

TABLE 4

True accuracy, estimated accuracies, and verified accuracies of an approximation corresponding to the linear system with matrix (3) *and right-hand side* b1. *LAPACK's GE routines were used for Estimations* 1 *and* 2.

Dimension	300	600	900	1200
True accuracy	14	14	14	14
Estimation 1	15	15	15	15
Estimation 2	12	12	12	12
Lowest verified	13	13	12	12
Highest verified	13	13	12	12

it is not possible to check the forward error. Nevertheless, it can be determined with a verifying routine as long as the iterative solver relies on a splitting of the data matrix [2].

Table 5 summarizes the results for different prescribed backward errors. (As before, we express errors in terms of accuracies.) The computing times normalized by the time needed to compute the approximation $\hat{x}$ are listed in Table 6. For low dimensions, verifying the error is twice as fast as the solving process, but for very high dimensions determining the error with a verifying routine can be as expensive as the computation of the approximation itself.

5. Further remarks. The quality of the delivered error bounds depends on the quality of the bound

$$|(LU)^{-1}| \leq \langle U\rangle^{-1}\langle L\rangle^{-1}, \tag{4}$$

TABLE 5

Prescribed accuracy, achieved accuracy, and verified accuracies of an approximation for a linear system of dimension 2916 *with matrix* (2) *and right-hand side* b2. *The approximation was computed with ITPACK's SSORSI routine.*

Prescribed accuracy	5	7	9	11	13
True accuracy	0	0	1	4	6
Lowest verified	0	0	1	2	5
Highest verified	6	9	11	12	14

TABLE 6

Runtime ratios solving a linear system with matrix (2) *and right-hand side* b2. *The approximation was computed with ITPACK's SSORSI routine.*

Dimension	100	2916	40000
Computation of $\hat{x}$	1	1	1
Verification	0.26	0.56	1.1

where $\langle\cdot\rangle$ denotes Ostrowski's comparison matrix[1] [2]. If $A = LU$ is an M-matrix[2] then equality holds in (4) and good results can be expected. For most H-matrices[3] (4) is still a good bound. Although many positive definite matrices (in particular symmetric M-matrices) are H-matrices, further research is needed to generalize the concept of verification to a wider class of matrices. Furthermore, an implementation of a verifying routine relies on the ability to manipulate the rounding direction and to check for floating-point exceptions.

Experiences with porting a library of verifying routines to another platform are reported in [5].

The C-programming code of the verifying routines can be obtained on the World Wide Web. At the time of this writing, the address is `http://www.ifi.unibas.ch/gruul/gruul.html`.

REFERENCES

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, LAPACK *Users' Guide*, 2nd ed., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1995.

[2] C. Falcó Korn, *Die Erweiterung von Software-Bibliotheken zur effizienten Verifikation der approximationslösung linearer Gleichungssysteme*, Ph.D. thesis, Institute for Computer Science, University of Basel, Switzerland, 1993.

[3] C. Falcó Korn and C. P. Ullrich, *Verified solution of linear systems based on common software libraries*, Interval Computations, 3 (1993), pp. 116–133.

[4] ———, *Extending LINPACK by verification routines for linear systems*, Math. Comput. Simulation, 39 (1995), pp. 21–37.

[5] ———, *LINPACK and ITPACK Extensions for Verified Computations—Interfaces, Usage, Portability*, Tech. report 94-2, Universitätsrechenzentrum und Institut für Informatik, Universität Basel, Switzerland, March 1994.

[6] D. R. Kincaid, J. R. Respess, D. M. Young, and R. G. Grimes, *ITPACK 2C: A FORTRAN package for solving large sparse linear systems by adaptive accelerated iterative methods*, ACM Trans. Math. Software, 3 (1982), pp. 302–322.

[7] G. Golub and C. F. Van Loan, *Matrix Computations*, 2nd. ed., The Johns Hopkins University Press, Baltimore, MD, 1989.

[8] F. Chatelin and V. Frayssé, *A Statistical Study of the Stability of Linear Systems*, Tech. report TR/PA/91/43, CERFACS, Toulouse, France, 1991.

[9] N. J. Higham, *Efficient algorithms for computing the condition number of a tridiagonal matrix*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 150–165.

[10] R. Barret et al., *Templates for the Solution of Linear Systems*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1994.

[1] The Ostrowski comparison matrix of a matrix $A = (a_{ij})$ is the matrix that has $|a_{ii}|$ for its (i, i) diagonal element and $-|a_{ij}|$ for its (i, j) off-diagonal element.

[2] A matrix $A = (a_{ij})$ is an M-matrix if $a_{ij} \leq 0$ for $i \neq j$, A is nonsingular, and $A^{-1} \geq 0$.

[3] A matrix is an H-matrix if its Ostrowski comparison matrix is an M-matrix.

SIAM J. SCI. COMPUT.
Vol. 17, No. 5, pp. 1019–1039, September 1996

HYBRID MULTIFLUID ALGORITHMS*

SMADAR KARNI†

Abstract. Extensions of many successful single-component schemes to compute multicomponent gas dynamics suffer from oscillations and other computational inaccuracies near material interfaces that are caused by the failure of the schemes to maintain pressure equilibrium between the fluid components. A new algorithm based on the compressible Euler equations for multicomponent fluids augmented by the pressure evolution equation is presented. The extended set of equations offers two alternative ways to update the pressure field: (i) using the equation of state or (ii) using the pressure evolution equation. In a numerical implementation, these two procedures generally yield different answers. The former is a standard conservative update, but may produce oscillations near material interfaces; the latter is nonconservative, but becomes exact near interfaces and automatically maintains pressure equilibrium. A hybrid scheme which selects from the two pressure update procedures is presented. The scheme perfectly conserves total mass and momentum and conserves total energy everywhere except at a finite (very small) number of grid cells. Computed solutions exhibit oscillation-free interfaces and have *negligible* relative conservation errors in total energy even for very strong shocks. The proposed hybrid approach and switching strategies are independent of the numerical implementation and may provide a simple framework within which to extend one's favourite scheme to solve multifluid dynamics.

Key words. Euler equations, compressible interfaces, shock-capturing

AMS subject classifications. 65M0G, 65M12, 65N15, 76T05

1. Introduction.

Within the shock-capturing numerical approach to solve the gas dynamics equations, flow discontinuities are not perfectly represented on a discrete grid. They take on diffused viscous-like profiles, typically occupying two to three grid cells for shock fronts and five to seven grid cells for contact surfaces. Problems posed by nonzero shock width have received attention in the literature in various computational contexts [7, 15, 16, 22]. In multifluid dynamics, the problem is not diffused shock layers but rather diffused material interfaces across which the fluid composition changes. From a computational aspect, this is a harder problem since shocks possess compressive mechanisms which tend to absorb any induced error back into the shock front. Interfaces do not possess such mechanisms, and it is easier for errors generated near or within the diffused front to escape and contaminate the solution. The transition across the interface is governed by numerical viscosity mechanisms and intermediate states are not necessarily physically consistent. Indeed, when extended to multicomponent fluids, many successful single-species schemes [9, 19, 23, 28] encounter difficulties maintaining pressure equilibrium among the fluid components across diffused interfaces [1, 12, 13, 14]. Consequently, updating the pressure field via the equation of state (EOS) often generates erroneous pressure fluctuations at material interfaces, which subsequently contaminate the solution of other flow variables. These oscillations are not the ones commonly associated with high-order schemes. They are present already in first-order schemes and are persistent. They not only render the solutions (at times fatally) oscillatory, but also set off dispersive acoustic mechanisms which tend to further thicken the interfaces. While exceptions may apply [2, 6], it is generally true that extending one's favourite single-species scheme to multispecies is likely to produce solutions that are plagued by oscillations and other computational inaccuracies [1, 12, 13, 14]. A phenomenon of similar flavour, which has received some appreciation within the aerodynamic community, is the occurrence of erroneous pressure fluctuations across numerically diffused shear interfaces.

*Received by the editors January 13, 1995; accepted for publication (in revised form) May 25, 1995.

†Courant Institute of Mathematical Sciences, 251 Mercer Street, New York, NY, 10012 (karni@cims.nyu.edu). The author is on leave from Temple University, Philadelphia, PA, 19122. This work was supported by an NSF Postdoctoral Fellowship, DOE contract DEFG0288ER25053, NSF grant DMS 94 96155, and ONR grant N00014-94-I-0525.

One possible cure is to employ algorithms which reconstruct the material front as a sharp discontinuity [3, 4, 17, 20], thus circumventing the problems arising due to interface smearing. This, however, comes at the expense of simplicity as such front-fitting/front-tracking algorithms raise nontrivial computational issues (e.g., keeping track of subgrid geometries for front reconstruction, net flux distribution on oddly shaped cells, stability restrictions due to the occurrence of small cells, etc.) and become quite involved to implement in multispace dimensions. A multifluid front-tracking algorithm has been developed in [6], based on the notion of volume-of-fluid (see for example [20] and references cited therein). The equations are written for the individual species, the jumps in the thermodynamic variables across the interface are tracked (the jump in tangential velocity is captured), and the interface geometry is reconstructed from the partial volumes of the fluid components and then evolved using the underlying single-fluid velocity field. Assuming isentropic compression effective, thermodynamic quantities are obtained for the fluid mixture which keep the fluid components in pressure equilibrium and also account for their possibly different compressibilities. This last property, while less critical for gas mixtures, could prove indispensable in gas/liquid interfaces where compressibilities of the individual components are widely different. This algorithm was used in [10] to study oblique shock refraction at a planar gas interface and has been applied to a more general EOS in [30].

In this paper we wish to remain within the front-capturing framework, particularly because of its attractive simplicity. We take a different approach, based on the instrumental observation that if the pressure field is evolved by using the pressure evolution equation, then pressure equilibrium among fluid components is automatically maintained and the generation of spurious pressure fluctuations is avoided. This observation has led the author to develop a scheme based on formulating the gas dynamics equations in terms of the so-called primitive variables [11, 12, 21]. Near material interfaces, the primitive formulation completely decouples into a set of linear advection equations for the density and the fluid marker variable (also the cross velocity components in higher space dimensions). Although generally nonconservative, near material fronts the model becomes *exact* and does not introduce conservation errors. Applications of interest, however, invariably involve shock waves near which conservation errors are introduced, manifested in incorrect shock speed and strength. By using a viscous perturbation approach, these conservation errors can be controlled and kept acceptably small for shocks of weak to moderate strengths (up to $M_s \approx 2.0$) [11, 12, 29]. Solutions thus produced have clean oscillation-free interfaces, and relative conservation errors are within 1%–2%.

It may be argued that sacrificing conservation completely for the sake of a primitive formulation is a drastic measure which discards the baby together with the bathwater. Indeed, the failure of schemes which are based on conservative flow models is local, predictable, and limited to the immediate vicinity of material fronts. Everywhere else, these schemes are perfectly adequate. In this paper, we present an elegant hybrid scheme, which switches between conservative and primitive updates of the pressure/energy fields. For simplicity, we focus on a two-component ideal gas model. Ideas certainly extend to more than two components in multispace dimension, and we believe also to nonideal cases. The latter may require some extra care in treating the multifluid thermodynamics. Work is already in progress for gas/liquid interfaces. We augment the standard two-component Euler system by the pressure evolution equation. The extended (degenerate) system offers two alternative ways to update the pressure field: (i) using the conserved variables via the EOS or (ii) using the pressure evolution equations. Numerically, the two procedures are equivalent to within numerical truncation error. Near flow discontinuities, however, they will give different answers. The former is a standard conservative solution update, which is good everywhere except possibly at material fronts where it might generate oscillations. The latter is nonstandard and nonconservative but, provided it is averaged in a thermodynamically consistent manner, becomes *exact* at interfaces and automatically maintains pressure equilibrium between the fluid components. We present a hybrid scheme that switches between the two pressure update procedures. Broadly speaking, the EOS is used either to update the pressure field from the total energy equation (conservative) or to update the total energy field from the pressure equation (nonconservative), depending on whether the grid cell is away from/near the material front. The resulting scheme conserves perfectly the total mass and momentum. The total energy is conserved everywhere except across a very small number of grid interfaces, namely those that lie within the material interface layer. In miscible multifluid models this may amount to as few as four to five cell interfaces. In immiscible models it may amount to only *one* (!), but is enough to prevent the pressure fluctuations from arising, thus preventing the oscillations in the rest of the flow variables which follow. Relative conservation errors in total energy are *negligible* (at less than 0.1% for reasonably fine grids) and decrease under mesh refinement. Furthermore, the errors seem not to be sensitive to the strengths of the shocks that are involved. The numerical results clearly demonstrate that the hybrid method is suitable for computing gas dynamics involving *very* strong shocks ($M_s \geq 100$) and in that respect constitutes a major improvement over the scheme in [11, 12]. Using the pressure evolution equation in this manner may be interpreted as imposing a thermodynamically consistent boundary condition on the moving material interface.

2. The equations. Consider the Euler system for two-component fluids, where ρ, u, p, and E are the density, velocity, pressure, and total energy of the fluid mixture:

$$\rho_t + (\rho u)_x = 0, \tag{1a}$$

$$(\rho u)_t + (\rho u^2 + p)_x = 0, \tag{1b}$$

$$E_t + (u(E + p))_x = 0, \tag{1c}$$

$$(\rho\psi)_t + (\rho\psi u)_x = 0, \tag{1d}$$

with the EOS for ideal gases

$$p = (\gamma(\psi) - 1)\left(E - \frac{1}{2}\rho u^2\right). \tag{2}$$

In the above equation, ψ represents a marker variable whose role is to identify the fluid composition. Various quantities can serve that purpose: ψ can represent the mass fraction of one of the species [13, 14, 12], in which case equation (1d) expresses species conservation. The resulting system conserves not only the total mass of the mixture but also the mass of the individual components. Alternatively, fluid state variables propagate with the fluid flow and obey an evolutionary equation of the form (1d). Such variables may also serve as marker variables (e.g., the specific heat ratio γ [25] or any function of it [2], or the heat release constant Q in chemically reactive flows [5]), in which case equation (1d) expresses passive advection of the state variable with the fluid velocity. These formulations conserve the total mass but may not strictly conserve the mass of the individual components. Another possibility is to take ψ to be a level-set function which, for example, measures the distance of a fluid particle from the material interface. The evolution of ψ in this case also obeys (1d) [18]. Note that although this choice has a front-tracking flavour, the overall algorithm remains a front-capturing algorithm and thus avoids the complexities associated with front tracking.

$\gamma = \gamma(\psi)$ is the effective specific heat ratio of the fluid mixture and depends naturally on the fluid composition. The specific functional form of $\gamma(\psi)$ depends on the assumptions of the model. If ψ denotes the mass fraction of one fluid component, then assuming pressure and thermal equilibrium yields an average γ based on specific heat constants [1, 13, 14]

$$\gamma(\psi) = \frac{\psi C_{v_1}\gamma_1 + (1-\psi)C_{v_2}\gamma_2}{\psi C_{v_1} + (1-\psi)C_{v_2}}. \tag{2a}$$

Assuming isentropic compression, the effective γ for a mixture of two fluid components in pressure equilibrium is an average based on specific volumes [6]:

$$\frac{1}{\gamma} = \frac{V}{\gamma_1} + \frac{1-V}{\gamma_2}, \tag{2b}$$

where V bears a simple relation to the level-set function $V = 1/(|\psi_l/\psi_r| + 1)$.

Note that the Euler model (1) does not account for mixing processes. Unless the fluid components are premixed, the only mixing that may occur is numerical mixing due to numerical diffusion induced by front-capturing schemes. The formulas for effective thermodynamics should thus be interpreted in that context. These formulas are more meaningful (probably also more critical) in the viscous case where genuine mixing takes place. Note further that exact solutions to (1) do not depend on the specific heat constants, hence neither should the numerical solution. The extent to which this happens in practice may serve to indicate the appropriateness of the formula for the effective thermodynamics.

We now augment system (1) by the pressure evolution equation. This equation is obtained by differentiating the EOS, which for ideal gases gives

$$p_t + up_x + \gamma p u_x = 0. \tag{1e}$$

We observe that if the fluid components are in pressure equilibrium, both p and u are constant (continuous) across the interface. If p is evolved using (1e), pressure remains constant (continuous) and equilibrium between fluid components is automatically preserved.

The augmented system (1a)–(1e) is, of course, degenerate and will be analyzed below. Here, we first note that the system offers two alternative ways to compute the pressure: (a) using the conserved variables (1a)–(1d) and the EOS (2) or (b) using the pressure evolution equation (1e). Numerically, the two procedures are equivalent to within numerical truncation error. Near flow discontinuities, however, they will generally give different answers and we can choose which answer we care to believe. The former is a standard conservative update which is suitable everywhere except possibly near material interfaces where pressure fluctuations may occur due to smearing of the interface. The latter is a nonconservative update, but it becomes *exact* near material fronts and automatically maintains pressure equilibrium. The algorithm we propose involves switching between the two update schemes for the pressure field, depending on whether the grid cell is near/away from a material interface. The switching strategy depends on the precise definition of the fluid marker variable ψ. Two simple switching strategies are discussed below which have proved effective.

The augmented system in matrix quasilinear form is

$$\begin{pmatrix} \rho \\ \rho u \\ E \\ \rho\psi \\ p \end{pmatrix}_t + \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ -u^2 & 2u & 0 & 0 & 1 \\ -uH & H & u & 0 & u \\ -u\psi & \psi & 0 & u & 0 \\ -ua^2 & a^2 & 0 & 0 & u \end{pmatrix} \begin{pmatrix} \rho \\ \rho u \\ E \\ \rho\psi \\ p \end{pmatrix}_x = 0, \tag{3}$$

where we use $a^2 = \sqrt{\gamma p/\rho}$ to denote the speed of sound and $H = \frac{a^2}{\gamma-1} + \frac{1}{2}u^2$ to denote the specific enthalpy. The characteristic polynomial of the matrix is

$$(\lambda - u)^3((\lambda - u)^2 - a^2) = 0,$$

revealing that the augmented system is hyperbolic, with a *triple* degeneracy in the particle field. The matrix of eigenvalues is

$$\Lambda = \operatorname{diag}(u - a,\ u,\ u,\ u,\ u + a) \tag{4}$$

and the corresponding complete set of eigenvectors is

$$\mathbf{r}_1 = \begin{pmatrix} 1 \\ u - a \\ H - ua \\ \psi \\ a^2 \end{pmatrix},\ \mathbf{r}_2 = \begin{pmatrix} 1 \\ u \\ \frac{1}{2}u^2 \\ \psi \\ 0 \end{pmatrix},\ \mathbf{r}_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix},\ \mathbf{r}_4 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix},\ \mathbf{r}_5 = \begin{pmatrix} 1 \\ u + a \\ H + ua \\ \psi \\ a^2 \end{pmatrix}. \tag{5}$$

Eigenvectors are defined to within a constant, and for consistency of physical dimensions the nonzero entry in $\mathbf{r}_3$ should have the dimensions of (velocity)^2. We leave this point for a moment and will return to it later. The eigenvectors $\{\mathbf{r}_k\}_{k=1}^5$ are linearly independent, and so a small change to the solution vector $\mathbf{W} = (\rho,\ \rho u,\ E,\ \rho\psi,\ p)^T$ has a unique representation as

$$\delta\mathbf{W} = \sum_{k=1}^{5} \alpha_k \mathbf{r}_k \tag{6}$$

for some wave strengths $\{\alpha_k\}_{k=1}^5$. The first, second, and fifth components of (6) involve only α_1, α_2, and α_5,

$$\alpha_1 + \alpha_2 + \alpha_5 = \delta\rho,$$

$$(\alpha_1 + \alpha_2 + \alpha_5)u + (-\alpha_1 + \alpha_5)a = \delta(\rho u),$$

$$(\alpha_1 + \alpha_5)a^2 = \delta p,$$

which, after using standard differential relations, have the recognizable solution

$$\begin{aligned} \alpha_1 &= \frac{\delta p - \rho a \delta u}{2a^2}, \\ \alpha_2 &= \frac{a^2 \delta\rho - \delta p}{a^2}, \\ \alpha_5 &= \frac{\delta p + \rho a \delta u}{2a^2}. \end{aligned} \tag{7a}$$

The fourth component in (6) can now be solved for α_4:

$$\alpha_4 = \rho\delta\psi. \tag{7b}$$

The fifth component in (6) gives

$$\alpha_3 = \delta E - \left(\frac{\delta p}{\gamma - 1} + \rho u \delta u + \frac{1}{2}u^2 \delta\rho\right).$$

Since pressure and energy changes are related through the EOS, we use (2) to simplify further, i.e.,

$$\tag{7c}\begin{aligned}\alpha_3 &= p\delta\left(\frac{1}{\gamma-1}\right)\\ &= -\frac{p}{(\gamma-1)^2}\gamma'(\psi)\delta\psi\\ &= -\frac{X}{\gamma-1}\rho\delta\psi\\ &= -\frac{X}{\gamma-1}\alpha_4,\end{aligned}$$

where we borrowed the notation $X = \frac{p}{(\gamma-1)\rho}\gamma'(\psi)$ from [14, 18, 12]. The degeneracy of the system is thus reflected in the fact that the third and fourth eigenfields can be combined into a single eigenfield

$$\mathbf{r}_{3,4} = \begin{pmatrix} 0 \\ 0 \\ -\dfrac{X}{\gamma-1} \\ 1 \\ 0 \end{pmatrix}, \qquad \alpha_{3,4} = \rho\delta\psi.$$

3. Switching strategy. Adopting a strategy to switch between the two procedures of updating the pressure field depends on the precise definition of the marker variable ψ. We have experimented with two flow models: (i) an immiscible level-set model [18] and (ii) a miscible mass-fraction model [13, 14, 27]. The models in [1, 25] are closely related. Both models take the form (1a)–(1d), with different definitions for the variable ψ. They have been thoroughly analyzed in [12] with respect to their properties for computing propagating interfaces and are briefly discussed below. The reader is referred to [13, 14, 18] for more details. We have extended these models by (1e) in the fashion described in the previous section and have found the following very simple switching strategies to be effective.

3.1. Level-set model. This is essentially the two-component immiscible Euler model proposed in [18]. In this model, the marker variable ψ is a level-set function. It is initialized so that at time $t = 0$ it is positive on one side of the interface (component 1) and negative on the other side (component 2) and is zero at the material interface separating the two components (as, for example, would be a function measuring the distance of a fluid particle from the interface). Since material interfaces propagate with the fluid velocity, it follows from equation (1d) that at any later time $t > 0$, the level set $\psi(x,t) = 0$ (which is a single point in this one-dimensional case) locates the material interface, and the condition $\psi < 0$ or $\psi > 0$ identifies which fluid component is being solved for. The ratio of specific heats, $\gamma(\psi)$, is then determined according to

$$\tag{8}\gamma(\psi) = \begin{cases}\gamma_1, & \psi > 0,\\ \gamma_2, & \psi < 0,\end{cases}$$

and the EOS (2) switches discontinuously across the interface. On a discrete mesh, one seeks the cell interface $(j + \frac{1}{2})$ satisfying

$$\psi_j \cdot \psi_{j+1} < 0 \,, \tag{9}$$

which indicates that the material interface is located somewhere between the grid point j and the grid point $j + 1$. Across this one cell interface, the pressure field is evolved by using the pressure evolution equation (1e). Everywhere else, (1e) is ignored and the pressure field is updated using the EOS (2) with the appropriate value of γ, depending on whether $\psi < 0$ or $\psi > 0$.

A slight variation of the above procedure gives a model which is consistent with the specific volume averaging for the effective γ (2b), where ψ is used to obtain the effective γ in the *one* cell occupied by both fluid components:

$$\frac{1}{\gamma(\psi)} = \frac{V}{\gamma_1} + \frac{1-V}{\gamma_2}, \quad V = \frac{1}{|\psi_j/\psi_{j+1}| + 1}.$$

3.2. Mass-fraction model. This model was used in [12, 13, 14, 27] (also [1, 2, 25]). The function ψ is the mass fraction of one gas component, and equation (1d) expresses species conservation. $\psi = 1$ or $\psi = 0$ indicate the presence of only one fluid component while intermediate values $0 < \psi < 1$ indicate a mixture. The gas components are assumed to be in thermal equilibrium. Using Dalton's law of partial pressures, the effective $\gamma(\psi)$ of the fluid mixture is

$$\gamma(\psi) = \frac{\psi C_{v_1}\gamma_1 + (1-\psi)C_{v_2}\gamma_2}{\psi C_{v_1} + (1-\psi)C_{v_2}}, \tag{10}$$

where C_{v_1} and C_{v_2} are the specific heat coefficients at constant volume for the respective fluid components. In this model, the interface layer is defined by the condition $0 < \psi < 1$ or, alternatively, by the condition $|\psi'| > 0$. On a discrete mesh, one identifies the cell interfaces where

$$|\psi_{j+1} - \psi_j| > 0. \tag{11}$$

Across the cell interfaces where the above condition is satisfied, the pressure field is evolved using the pressure evolution equation (1e). Everywhere else, the EOS (2)+(10) is used. Note that in the latter situations ψ is very close to either 0 or 1, and $\gamma(\psi)$ automatically takes the appropriate value of either γ_1 or γ_2. In practice, one uses the condition $|\psi_{j+1} - \psi_j| > \epsilon$, where $\epsilon > 0$ is a small number (typically between 0.01 and 0.05), to ensure that only genuine interface layers are identified.

4. Numerical results. The hybrid approach described above is independent of the details of the numerical implementation. Thus, it may provide a general framework within which to extend one's favourite scheme to multicomponent flow computations and yield a scheme which is capable of producing clean oscillation-free interfaces as well as handling complicated interactions involving *very* strong shocks.

4.1. The scheme. We have implemented a Roe-type second-order upwind scheme [23, 14, 18]. Using the *fluctuation-and-signal* approach [24], the scheme lends itself to a convenient and uniform treatment of both the conservative (1a)–(1d) and nonconservative (1e) parts of the system. When restricted to only the conservative part, the scheme has an equivalent flux formulation. There is, of course, no flux formulation for the solution of the pressure evolution equation (1e), since pressure is not a conserved quantity. As with the standard Roe method, the system is locally linearized about an average state. Flow gradients are projected onto the linearized eigenvectors

$$\mathbf{r}_1 = \begin{pmatrix} 1 \\ \tilde{u} - \tilde{a} \\ \tilde{H} - \tilde{u}\tilde{a} \\ \tilde{\psi} \\ \tilde{a}^2 \end{pmatrix}, \quad \mathbf{r}_2 = \begin{pmatrix} 1 \\ \tilde{u} \\ \frac{1}{2}\tilde{u}^2 \\ \tilde{\psi} \\ 0 \end{pmatrix}, \quad \mathbf{r}_3 = \begin{pmatrix} 0 \\ 0 \\ -\dfrac{\tilde{X}}{\tilde{\gamma} - 1} \\ 1 \\ 0 \end{pmatrix}, \quad \mathbf{r}_4 = \begin{pmatrix} 1 \\ \tilde{u} + \tilde{a} \\ \tilde{H} + \tilde{u}\tilde{a} \\ \tilde{\psi} \\ \tilde{a}^2 \end{pmatrix} \tag{12}$$

with wave strengths

$$\begin{aligned} \alpha_1 &= \frac{\Delta p - \tilde{\rho}\tilde{a}\Delta u}{2\tilde{a}^2}, \\ \alpha_2 &= \frac{\tilde{a}^2\Delta\rho - \Delta p}{\tilde{a}^2}, \\ \alpha_3 &= \tilde{\rho}\Delta\psi, \\ \alpha_4 &= \frac{\Delta p + \tilde{\rho}\tilde{a}\Delta u}{2\tilde{a}^2} \end{aligned} \tag{13}$$

and are propagated by a limited Lax–Wendroff marching step [26] with linearized wave speeds

$$\begin{aligned} \lambda_1 &= \tilde{u} - \tilde{a}, \\ \lambda_2 &= \tilde{u}, \\ \lambda_3 &= \tilde{u}, \\ \lambda_4 &= \tilde{u} + \tilde{a}. \end{aligned} \tag{14}$$

Note that due to the degeneracy of the system, we have in this case *four* eigenfields, each of which is a *five*-component vector.

The Roe averages satisfy the property $\tilde{A}(\mathbf{W}_L - \mathbf{W}_R) = \mathbf{F}_L - \mathbf{F}_R$ for any two states $\mathbf{W}_L$ and $\mathbf{W}_R$ and the corresponding flux functions $\mathbf{F}_L$ and $\mathbf{F}_R$. For the two models described above, the average values are calculated from the conservative part of the governing equations (1a)–(1d). This guarantees that away from the material interface the method reduces to the standard conservative Roe scheme [23, 24]. The averages are given by

$$\begin{aligned} \tilde{\rho} &= \sqrt{\rho_L \rho_R}, \\ \tilde{u} &= \frac{\sqrt{\rho_L}u_L + \sqrt{\rho_R}u_R}{\sqrt{\rho_L} + \sqrt{\rho_R}}, \\ \tilde{H} &= \frac{\sqrt{\rho_L}H_L + \sqrt{\rho_R}H_R}{\sqrt{\rho_L} + \sqrt{\rho_R}}, \\ \tilde{\psi} &= \frac{\sqrt{\rho_L}\psi_L + \sqrt{\rho_R}\psi_R}{\sqrt{\rho_L} + \sqrt{\rho_R}}, \\ \tilde{a}^2 &= (\gamma(\tilde{\psi}) - 1)\left(\tilde{H} - \frac{1}{2}\tilde{u}^2\right). \end{aligned} \tag{15a}$$

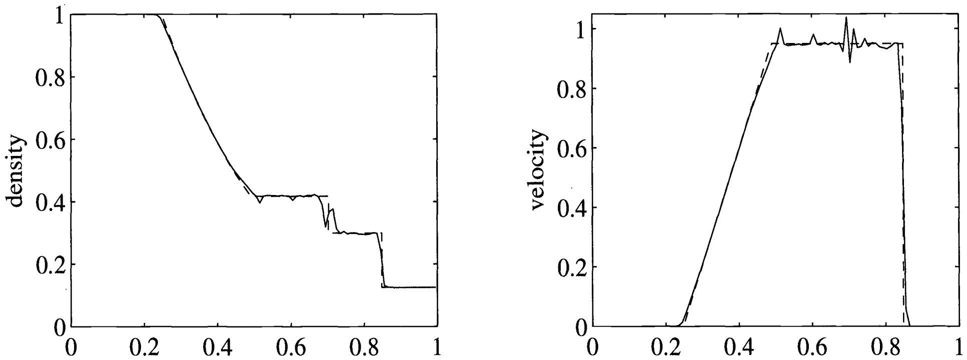


FIG. 1a. *Two-component shock-tube problem computed by the level-set model in* [18]. *Computed* (*solid*) *and exact* (*dash*) *solutions.*

For the level-set model (see [18] for details),

$$\tilde{X} = \frac{\Delta p - (\gamma(\tilde{\psi}) - 1)\Delta \frac{p}{\gamma - 1}}{\tilde{\rho}\Delta\psi}. \tag{15b}$$

For the mass-fraction model (see [13, 14] for details),

$$\begin{aligned} \tilde{X} &= \frac{C_{v_1}C_{v_2}}{\tilde{C}_v}\tilde{e}(\gamma_1 - \gamma_2), \\ \tilde{e} &= \frac{\sqrt{\rho_L}e_L + \sqrt{\rho_R}e_R}{\sqrt{\rho_L} + \sqrt{\rho_R}}, \\ \tilde{C}_v &= C_{v_1}\tilde{\psi} + C_{v_2}(1 - \tilde{\psi}), \end{aligned} \tag{15c}$$

where $e = \frac{p}{\rho(\gamma-1)}$ is the internal energy and $\gamma(\tilde{\psi})$ is given by equation (10). The pressure field is updated using the EOS (2), while ignoring the fifth component of the solution vector $\mathbf{W}$ everywhere in the flow field except across cell interfaces where either condition (9) or (11) holds. Across those cell interfaces, the pressure field is evolved via the fifth component of the solution algorithm by using the same limited Lax–Wendroff marching scheme with the same conservative linearization (15). The amount of extra work required for the nonstandard update of the pressure field is *negligible* since all the information needed (sound speed, wave strengths, etc.) is already available. Once the new pressure field is calculated, the new energy field E is computed using (2). This step is not strictly conservative (work is currently in progress to modify this step so that it is conservative).

4.2. The experiments.

Test A. The initial data correspond to a two-component shock-tube problem, also used in [12, 13, 27]:

$$\mathbf{W}_L = (\rho_L,\ u_L,\ p_L,\ \gamma_L) = (1.0,\quad 0.0,\ 1.0,\ 1.4),$$

$$\mathbf{W}_R = (\rho_R,\ u_R,\ p_R,\ \gamma_R) = (0.125,\ 0.0,\ 0.1,\ 1.2).$$

Figure 1 shows solutions based on the conservative flow models in [13, 14, 18] and by the present hybrid scheme. In these computations we used 200 grid points and a Courant–Friedrichs–Lewy (CFL) number of 0.9. For the second-order calculations we used the superbee flux limiter [26]. Strong oscillations plague the solution obtained by the level-set algorithm of [18] near the material interface (Figure 1a). These oscillations are visible in all

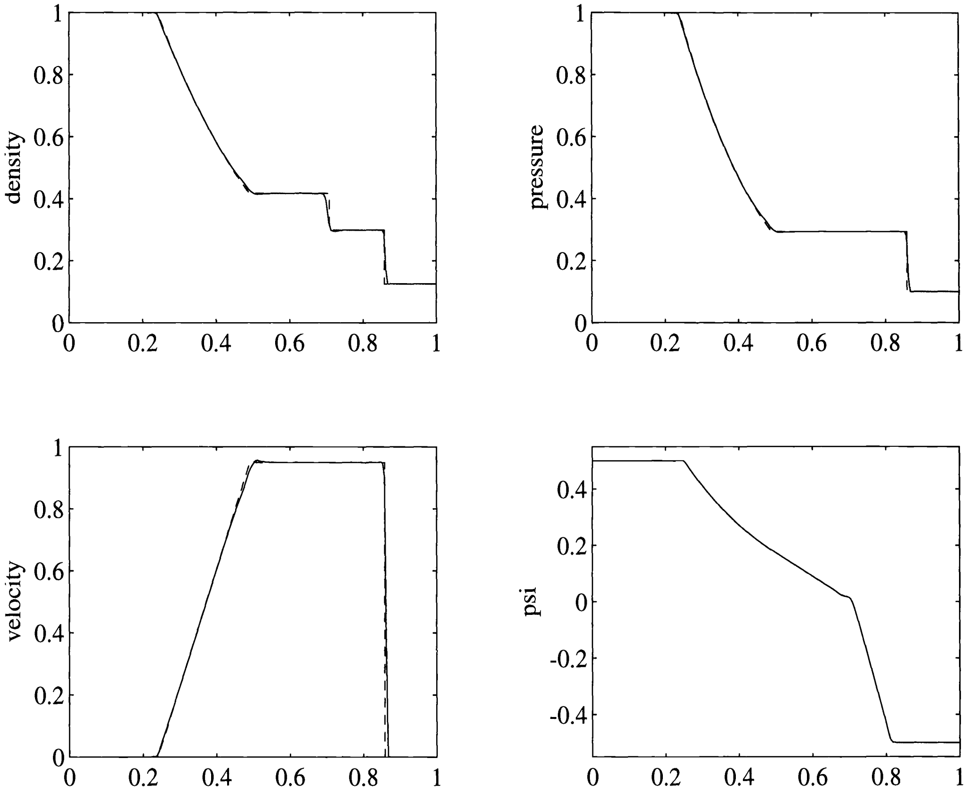


FIG. 1b. *Same as Figure* 1a, *computed by the hybrid level-set scheme* (12)–(14), (15a), (15b). *Nonconservative solution update across one cell interface. Computed* (*solid*) *and exact* (*dash*) *solutions.*

flow variables. (Interestingly, the only nonoscillatory variable is the fluid marker function ψ which is obtained by computing at each grid point $(\rho\psi)_j/(\rho)_j$, both of which are apparently oscillating in phase, see [12].)

The oscillations are completely eliminated from the solution by switching to a nonconservative pressure update at the material interface using the pressure evolution equation (1e) (Figure 1b). This pressure update is used across only *one* (!) cell interface, but is sufficient to prevent the generation of pressure fluctuations at the interface, thereby preventing the subsequent oscillations in the other flow variables. Comparison between the computed and the exact solution (also included in the figure) shows excellent agreement, although the scheme is not strictly conservative. With this hybrid algorithm, the total mass and momentum are perfectly conserved but the total energy is not. For the calculation shown in Figure 1b, the relative error in total energy is 0.242% on a 200-point mesh. Repeating the same test with 800 mesh points yields relative conservation error of 0.066% in the total energy.

The solution obtained by the mass-fraction model of [13, 14] exhibits a nonphysical velocity step across the material front and a density undershoot (hence, a temperature overshoot). Both are present already in a first-order computation (velocity profile, Figure 1c), with the density undershoot becoming more pronounced in a second-order computation (density profile, Figure 1c); see also [1, 13, 12]. Interestingly, the computed pressure field itself (not shown) appears to be well behaved. The computational inaccuracies are most apparent in the velocity and density profiles. Nonetheless, the cause for this false behaviour is the initial erroneous pressure fluctuations that disturb the pressure equilibrium. Indeed, taking measures

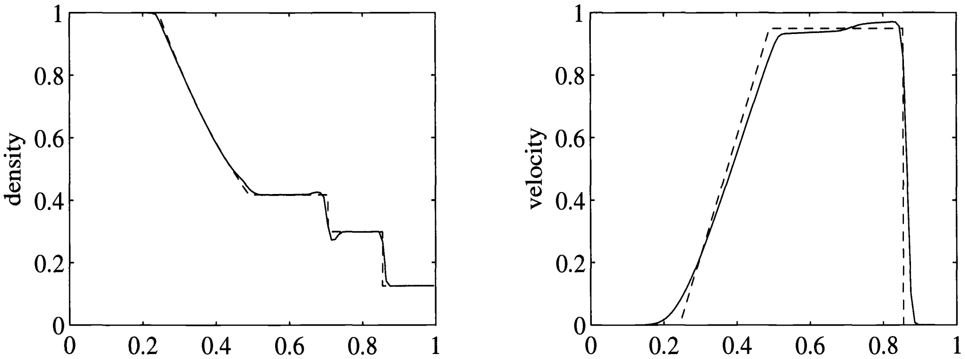


FIG. 1c. *Two-component shock-tube problem computed by the mass-fraction model in* [13, 14]. *Computed* (*solid*) *and exact* (*dash*) *solutions.*

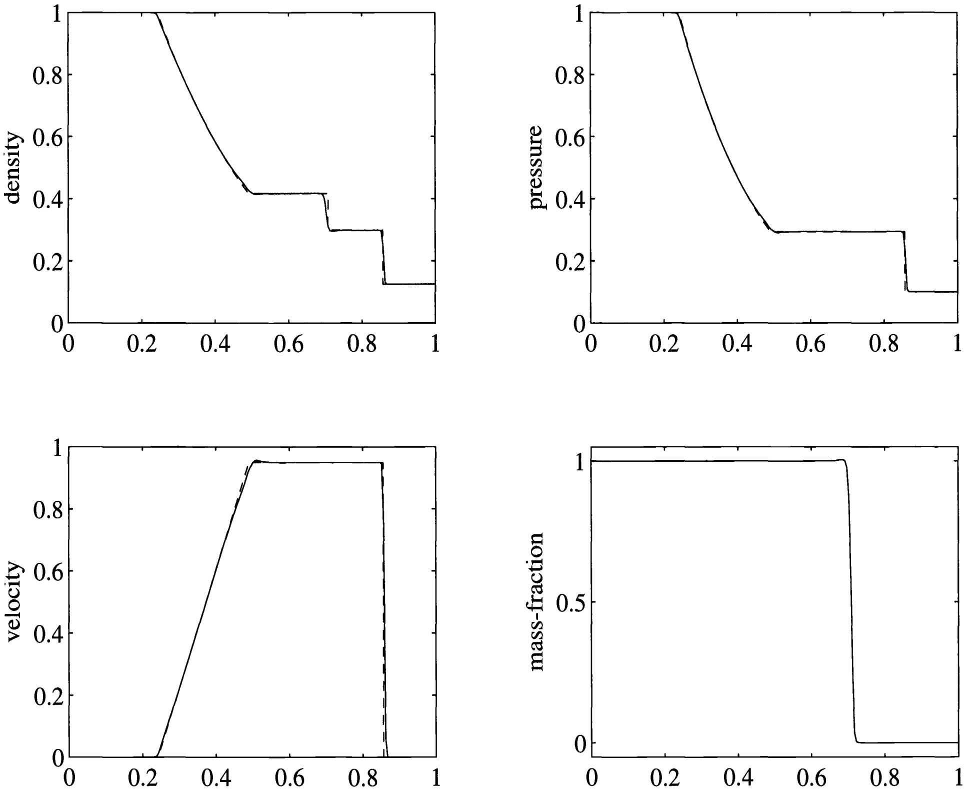


FIG. 1d. *Same as Figure* 1c, *computed by the hybrid mass-fraction scheme* (12)–(14), (15a), (15c). *Nonconservative solution update across four to five cell interfaces. Computed* (*solid*) *and exact* (*dash*) *solutions.*

to prevent the pressure fluctuations from arising completely removes the false behaviour from the density and velocity fields (Figure 1d). Using (1e) only at cell interfaces where condition (11) is satisfied (approximately four to five cell interfaces with $\epsilon = 0.05$) is sufficient to prevent the generation of pressure fluctuations and the computational inaccuracies in the other flow variables which follow (Figure 1d). The computed solution shows excellent agreement with the exact solution (also included in the figures) although the computation is not strictly

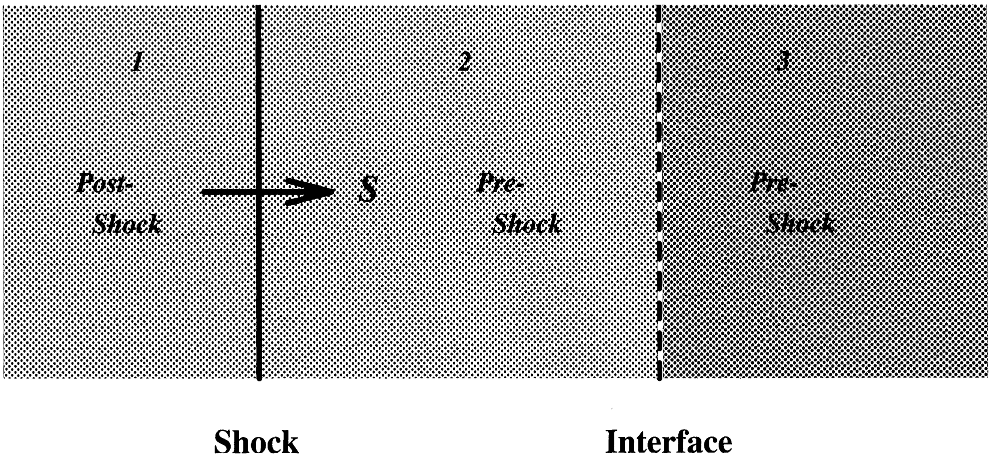


FIG. 2. *Shock wave impinging on a material interface.*

conservative. Here again, the total mass (also the mass of the individual species) and momentum are perfectly conserved, and relative conservation error in the total energy is negligible at 0.054% for the calculation shown in Figure 1d. If the same calculation is repeated with 800 grid points, the relative conservation error decreases further to 0.026%.

Test B. The initial data correspond to a weak shock wave in air ($M_s = 1.1952$), refracting at a helium material interface (see Figure 2):

$$\mathbf{W}_1 = (\rho_1,\ u_1,\ p_1,\ \gamma_1) = (1.3333,\ 0.3535,\ 1.5,\ 1.4), \quad \text{air, postshock;}$$

$$\mathbf{W}_2 = (\rho_2,\ u_2,\ p_2,\ \gamma_2) = (1.0,\quad 0.0,\ 1.0,\ 1.4), \quad \text{air, preshock;}$$

$$\mathbf{W}_3 = (\rho_3,\ u_3,\ p_3,\ \gamma_3) = (0.1379,\ 0.0,\ 1.0,\ 1.67), \quad \text{helium, preshock.}$$

In shock refraction problems like this, upon being hit by the incident shock wave, the interface is set into motion. A shock wave is transmitted across the interface into the receiving gas, which travels at a higher or lower speed than that of the incident shock depending on the respective speed of sound. In geometrical acoustics, the ratio of incident/transmitted shock speeds is often called the *index of refraction.* Depending on the so-called acoustic impedance of the receiving gas, the reflected wave is either a shock wave or an expansion [8, 10]. While playing a less apparent role in one-dimensional shock refraction problems, both the index of refraction and the acoustic impedance are crucial parameters in two-dimensional shock refraction at oblique/curved interfaces, dictating the sign and amount of vorticity generation at the interface, hence the characteristics of global dynamics [21] and the appearance of regular and irregular shock refraction patterns (see [10, 21] and references cited therein). In such problems, the material interface is also a physically unstable shear interface, and any erroneous numerical behaviour may trigger completely false dynamics. In this air/helium case, the reflected wave is a very narrow rarefaction. The transmitted shock wave is very weak. As the sound speed in helium is higher than that in air, the weak transmitted shock wave travels faster than the incident shock.

Figure 3a shows the computed solution by the level-set model. As soon as the incident shock wave hits the interface and sets it into motion, strong oscillations appear in the solution. Once generated, the oscillations disperse and appear to follow behind the transmitted shock front. As in the previous shock-tube test, switching to a pressure update based on the pressure evolution equation (1e) across only one cell interface prevents the generation of the initial

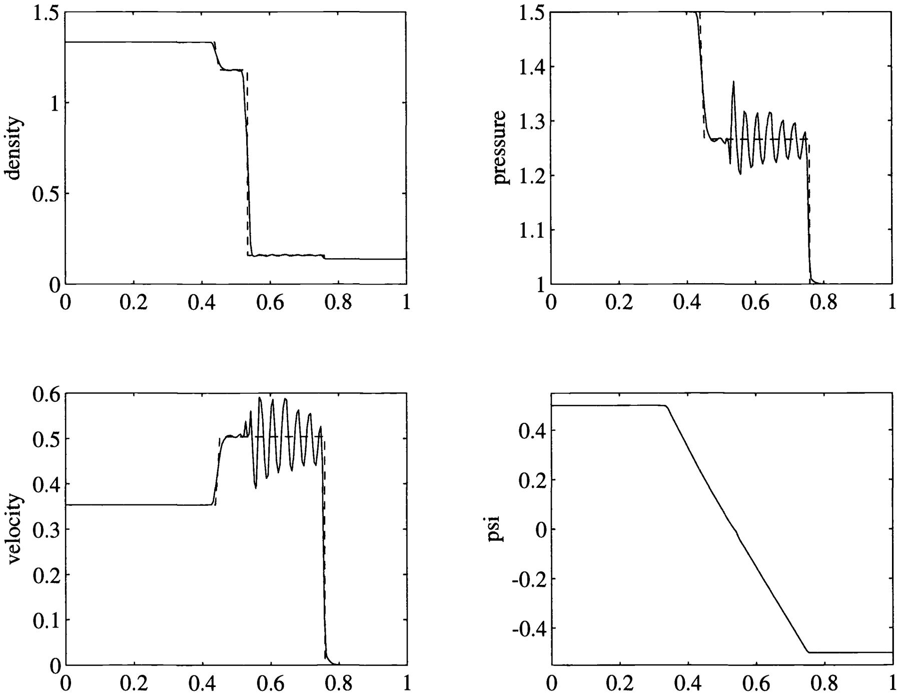


FIG. 3a. *Shock wave in air, $M_s = 1.1952$, refracting at a helium interface computed by the level-set model in* [18]. *Computed* (*solid*) *and exact* (*dash*) *solutions.*

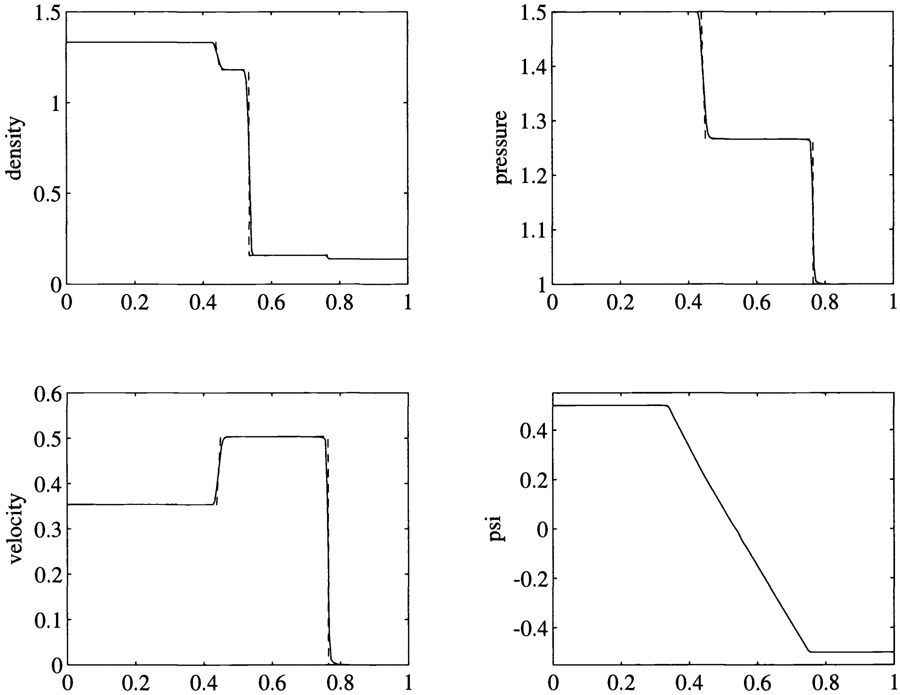


FIG. 3b. *Same as Figure* 3a, *computed by the hybrid level-set scheme* (12)–(14), (15a), (15b). *Nonconservative solution update across one cell interface. Computed* (*solid*) *and exact* (*dash*) *solutions.*

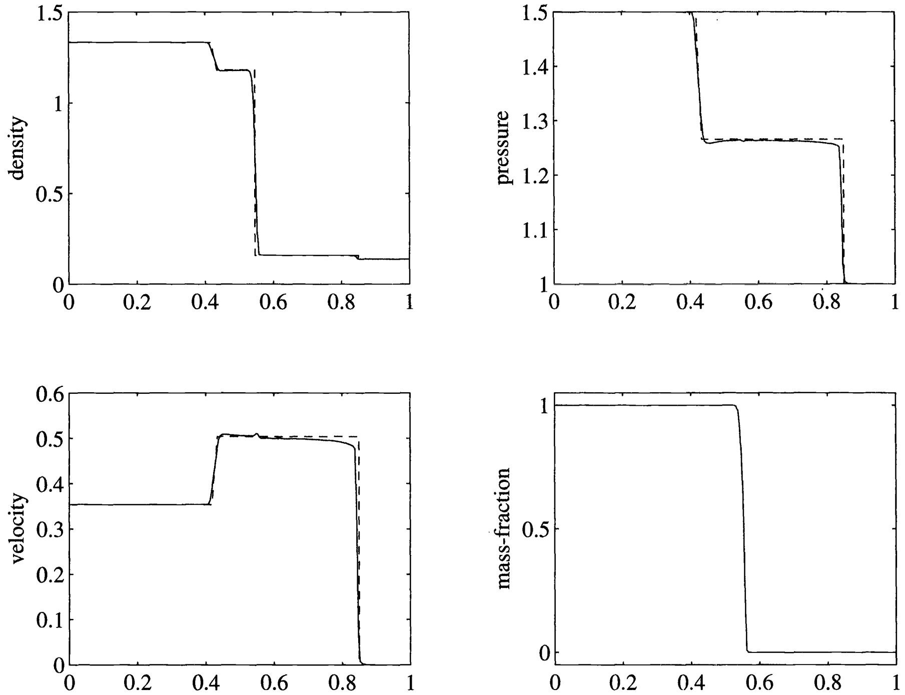


FIG. 3c. *Shock wave in air, $M_s = 1.1952$, refracting at a helium interface computed by the mass-fraction model in* [13, 14]. *Computed (solid) and exact (dash) solutions.*

pressure fluctuation and is sufficient to completely eliminate the oscillations from the solution (Figure 3b). The agreement between the computed and exact solutions is excellent.

The computed solution obtained by the mass-fraction model (Figure 3c) also exhibits a nonphysical behaviour. The velocity profile fails to maintain uniform velocity in the region between the transmitted shock and the tail of the reflected fan. The pressure profiles are also visibly nonuniform. Updating the pressure field via equation (1e) across as few as four to five cell interfaces is sufficient to eliminate these problems (Figure 3d). The agreement between the computed and exact solutions is again excellent.

Test C. We present another example of a weak shock wave in air ($M_s = 1.1952$) refracting at a material interface, with in this case a heavy gas refrigerant R22 (commercially known as freon). The initial data are

$$\mathbf{W}_1 = (\rho_1,\ u_1,\ p_1,\ \gamma_1) = (1.3333,\ 0.3535,\ 1.5,\ 1.4), \quad \text{air, postshock;}$$

$$\mathbf{W}_2 = (\rho_2,\ u_2,\ p_2,\ \gamma_2) = (1.0,\ \ 0.0,\ 1.0,\ 1.4), \quad \text{air, preshock;}$$

$$\mathbf{W}_3 = (\rho_3,\ u_3,\ p_3,\ \gamma_3) = (3.1538,\ 0.0,\ 1.0,\ 1.249), \quad \text{R22, preshock.}$$

The transmitted shock speed in R22 is lower than that of the incident shock wave in air, and the reflected wave is a weak shock. Results are shown for the level-set model (Figure 4a) and the hybrid level-set version (Figure 4b). The former solution is contaminated by strong oscillations. As in the previous tests, switching to a pressure update via (1e) across only one cell interface is sufficient to remove completely the oscillations from the computed level-set solution. The agreement between the computed and the exact solutions is excellent.

Test D. The data in this test correspond to a stronger shock in air ($M_s = 3.6055$) refracting on helium (Figure 5a) and R22 (Figure 5b) material interfaces, respectively. In both cases,

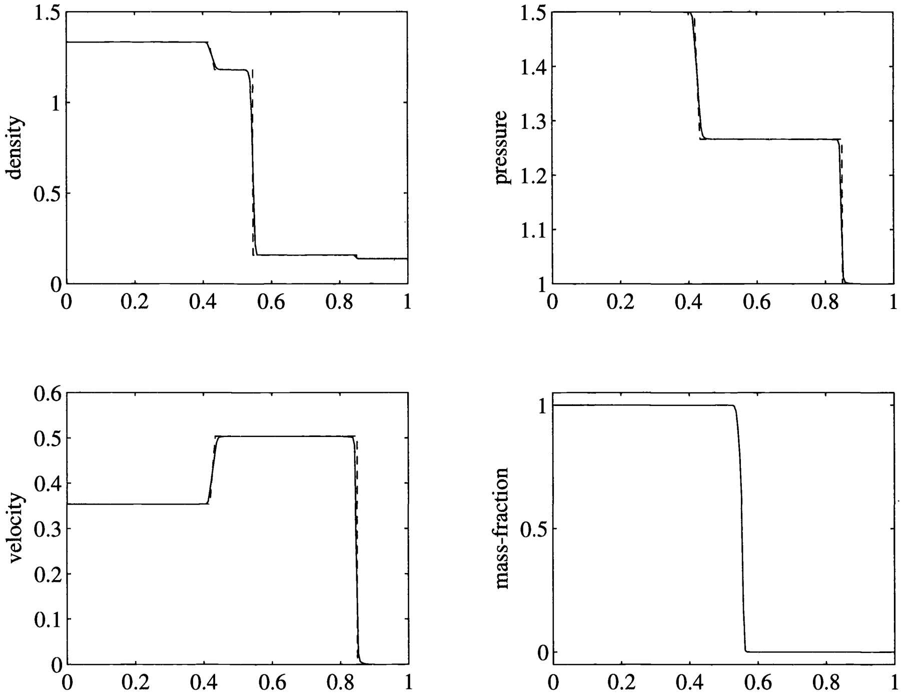


FIG. 3d. *Same as Figure* 3c, *computed by the hybrid mass-fraction scheme* (12)–(14), (15a), (15c). *Nonconservative solution update across four to five cell interfaces. Computed* (*solid*) *and exact* (*dash*) *solutions.*

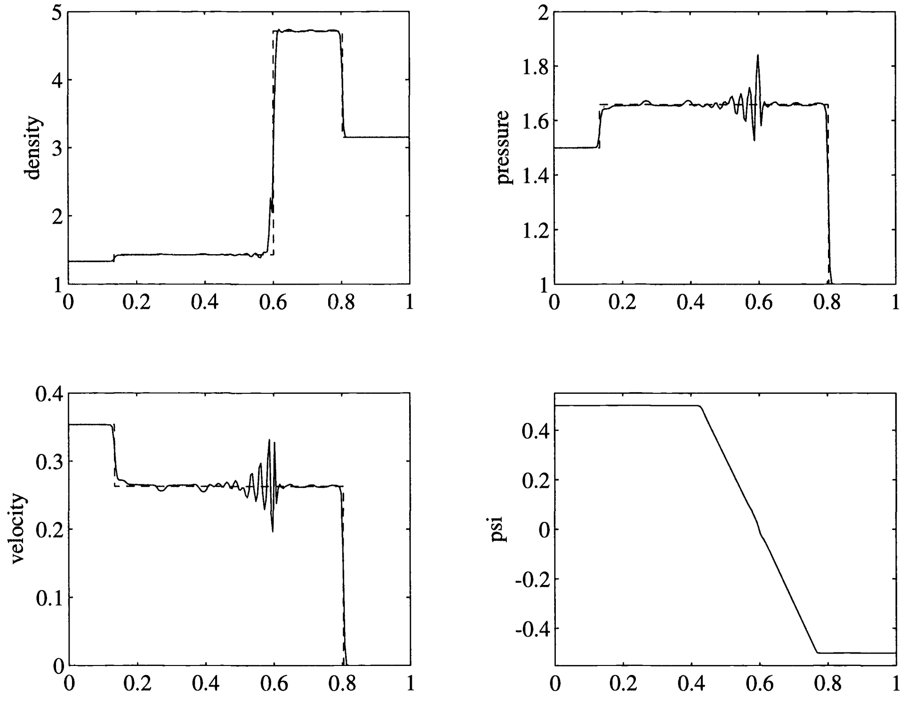


FIG. 4a. *Shock wave in air, M_s = 1.1952, refracting at an* R22 *interface computed by the level-set model in* [18]. *Computed* (*solid*) *and exact* (*dash*) *solutions.*

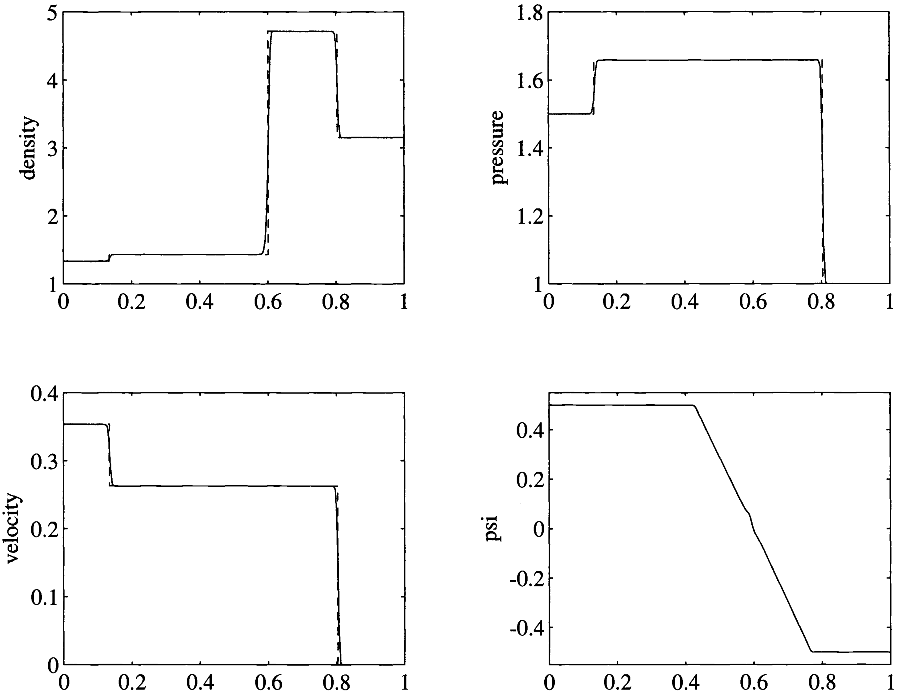


FIG. 4b. *Same as Figure* 4a, *computed by the hybrid level-set scheme* (12)–(14), (15a), (15b). *Nonconservative solution update across one cell interface. Computed* (*solid*) *and exact* (*dash*) *solutions.*

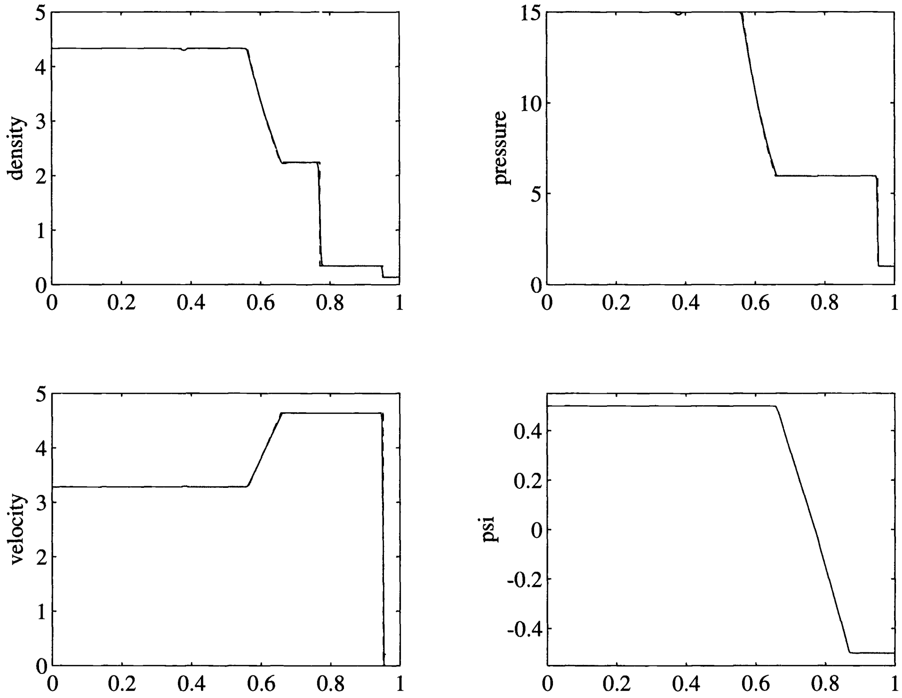


FIG. 5a. *Stronger shock wave in air,* $M_s = 3.6055$, *refracting at a helium interface computed by the hybrid level-set scheme* (12)–(14), (15a), (15b). *Computed* (*solid*) *and exact* (*dash*) *solutions.*

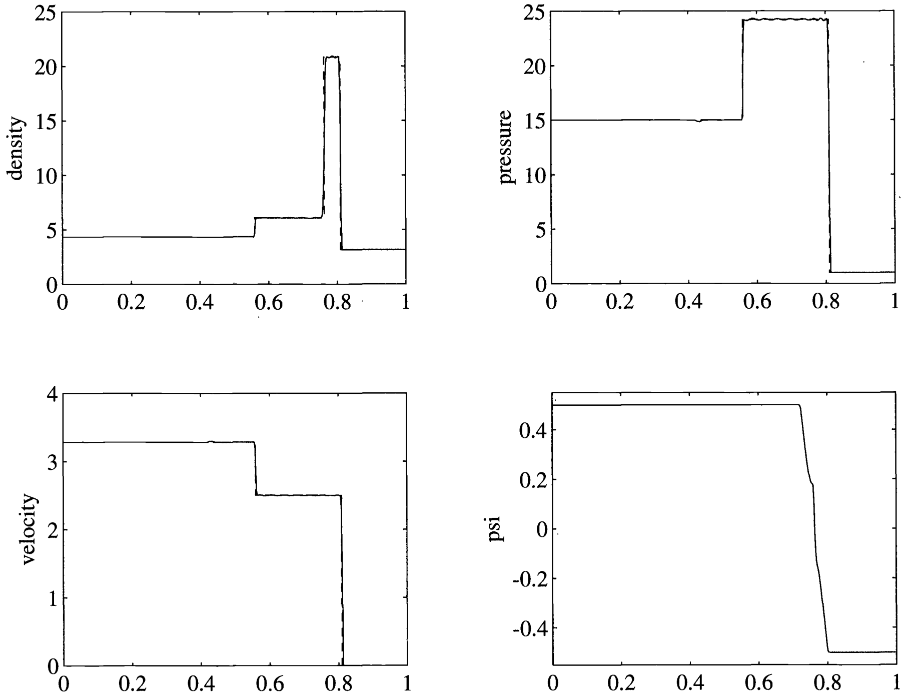


FIG. 5b. *Stronger shock wave in air, M_s = 3.6055, refracting at an* R22 *interface computed by the hybrid level-set scheme* (12)–(14), (15a), (15b). *Computed* (*solid*) *and exact* (*dash*) *solutions.*

due to the high velocity driving the incident shock, the entire flow field is supersonic and all the waves (including the reflected waves) are rightgoing. The incident shock as well as the transmitted and reflected waves are now stronger (compare with Figures 3 and 4). The aim in this test is to demonstrate the capability of the hybrid scheme to handle well interactions involving strong shocks.

Results with the hybrid level-set scheme are shown in Figure 5 for both a helium interface in which the refracted wave is a rarefaction, and for an R22 interface, in which the reflected wave is a shock. The preshock states for the air/helium case (Figure 5a) are the same as in Test B. The postshock state is

$$\mathbf{W}_1 = (\rho_1,\ u_1,\ p_1,\ \gamma_1) = (4.3333,\ 3.2817,\ 15.0,\ 1.4), \quad \text{air, postshock.}$$

The preshock states for the air/R22 case (Figure 5b) are the same as in Test C. The postshock state is the same as $\mathbf{W}_1$ above. The agreement between the computed and the exact solutions is excellent in both cases, and the relative conservation errors in total energy are negligible. For example, for the air/helium case the relative conservation error in total energy at time $T = 0.1169$ is 0.1301% with 400 grid points (0.0621% with 800 grid points). For the air/R22 case, the relative conservation error at time $T = 0.1637$ is 0.2249% with 400 grid points (0.1729% with 800 grid points). Table 1 summarizes a convergence study of the relative energy conservation errors for an $M_s = 9.2659$ shock wave in air impinging on a helium and an R22 interface, respectively (errors are measured against exact solutions). The errors are extremely small and decrease under mesh refinement at what appears to be a linear rate. These results clearly demonstrate that the present hybrid method is capable of handling well wave interactions involving strong shocks.

TABLE 1

Relative conservation errors in total energy for a strong shock wave in air ($M_s = 9.2659$) impinging on helium and R22 *interfaces.*

No. of points	Relative error in %
air/helium	(final time $T = 4.81 \times 10^{-2}$)
400	0.060
800	0.035
1600	0.021
air/R22	(final time $T = 6.41 \times 10^{-2}$)
400	0.084
800	0.048
1600	0.030

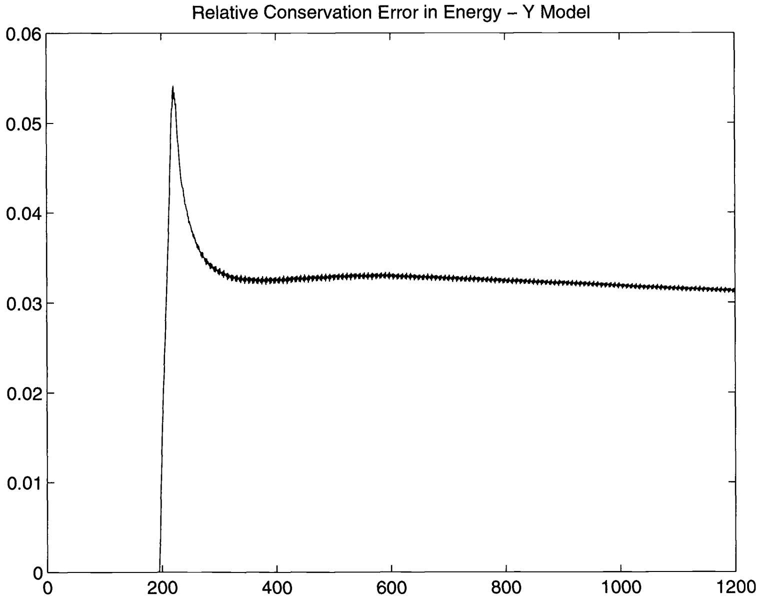


FIG. 6. *Relative conservation error in total energy as a function of number of time steps for a very strong shock wave in air, $M_s = 113.4$, refracting at a helium interface. Computed solution by the hybrid mass-fraction scheme* (12)–(14), (15a), (15c).

Test E. It is interesting to see where conservation error is generated and how it manifests itself. The data for the present test correspond to a very strong shock, $M_s = 113.4$, hitting an air/helium interface. Figure 6 shows relative conservation error in total energy as a function of the number of time steps. The error is obtained by comparing the results from a conservative and a hybrid computation:

$$100 * \frac{|(\sum E_j)_{\text{hybrid}} - (\sum E_j)_{\text{conservative}}|}{(\sum E_j)_{\text{conservative}}}.$$

From Figure 6 we observe that conservation error is generated only during wave interaction, namely, when the shock wave hits the interface. During the first couple hundred time steps, there is no conservation error. Error is then generated and reaches a peak value during the interaction, which then relaxes toward a constant value. No additional error is generated after the interaction is complete, which seems to indicate that waves propagate at the correct

TABLE 2

Relative conservation error in total energy as a function of number of time steps for a strong shock wave in air ($M_s = 113.4$) *impinging on a helium interface.*

No. of points	Relative error in %
200	0.1445
400	0.0788
800	0.0418
1600	0.0226

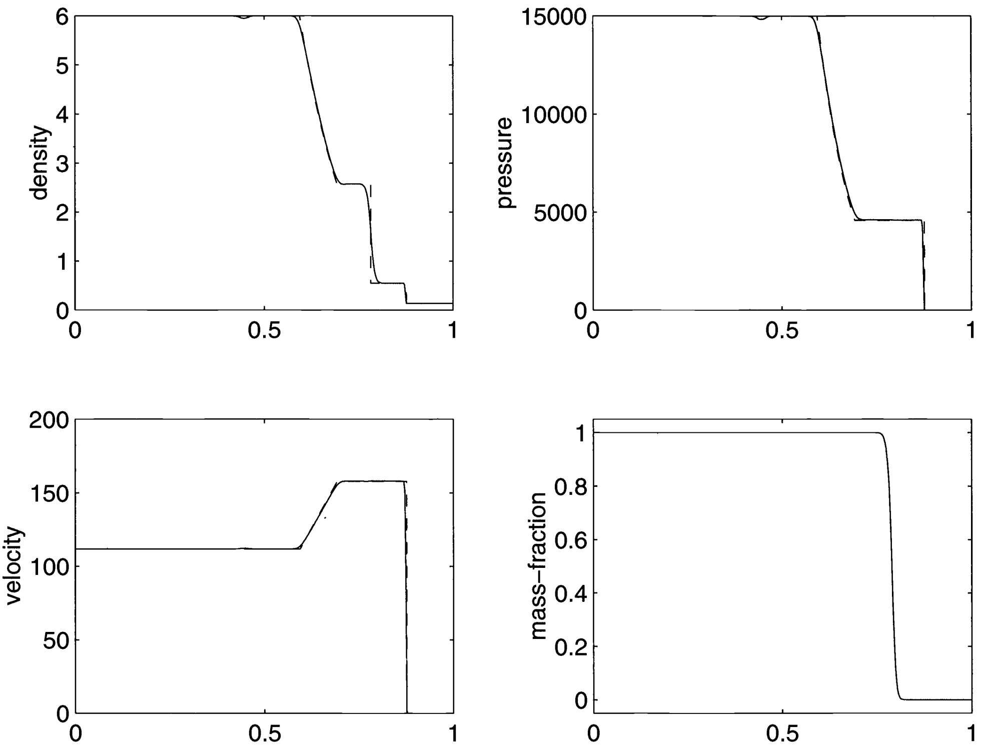


FIG. 7. *A very strong shock wave in air,* $M_s = 113.4$, *refracting at a helium interface computed by the hybrid mass-fraction scheme* (12)–(14), (15a), (15c). *Computed* (*solid*) *and exact* (*dash*) *solutions.*

speed. In one-dimensional setups, the generation of errors is a one-off event. In higher dimensions, for example, in shock-bubble interactions, errors may be provoked more commonly. In view of the smallness of the relative error, this issue does not seem to be a concern.

Table 2 gives relative errors in a sequence of runs with increasingly fine grids from which it is clear that (i) the solution seems to converge to the right place, (ii) convergence appears to be linear, and (iii) conservation error is *very* small and seems not to be sensitive to the strength of the shock involved. Actual results for this test are shown in Figure 7.

5. Conclusions. A hybrid approach has been presented for computing the dynamics of compressible multicomponent fluids based on augmenting the standard multicomponent gas dynamics equations by the pressure evolution equation. The extended system offers two choices for updating the pressure field: (i) a conservative update using the EOS (2) used throughout the flow field except near material interfaces where it may produce oscillations and (ii) a nonconservative update using the pressure evolution equation (1e) used near interfaces to ensure monotone solutions. A simple switching strategy between the two pressure update procedures has been presented (i) for an immiscible flow model based on a level-set formulation [18] and (ii) for a miscible flow model based on a mass-fraction formulation [1, 13, 14, 25, 27]. The resulting hybrid algorithm conserves the total mass and momentum of the system. The total energy is not perfectly conserved but relative conservation errors are *negligible* (of the order of a fraction of a percent). The hybrid strategy has proved to be very effective in computing shock-interface interactions involving shocks ranging from $M_s \approx 1.1$ to $M_s \approx 100$. Both the hybrid approach and the switching strategies are independent of the numerical implementation and may provide a convenient framework within which to extend one's favourite single-component scheme to multicomponents.

Acknowledgments. I would like to thank Mike Minion who read an earlier version of this manuscript and was more than happy to do to me what Alex Chorin had done to him. His comments were invaluable and are gratefully acknowledged. I would also like to thank Marsha Berger and Vinh Ton for their observant and constructive comments.

REFERENCES

[1] R. Abgrall, *Generalisation of the Roe scheme for the computation of mixture of perfect gases*, Rech. Aérospat., 6 (1988), pp. 31–43.

[2] ———, Private communication, 1994.

[3] I.-L. Chern, J. Glimm, O. McBryan, B. Plohr, and S. Yaniv, *Front tracking for gas dynamics*, J. Comput. Phys., 62 (1986), pp. 83–110.

[4] I. L. Chern and P. Colella, *A Conservative Front Tracking Method for Hyperbolic Conservation Laws*, Technical Report UCRL-97200, Lawrence Livermore National Laboratory, Livermore, CA, July 1987.

[5] J. F. Clarke, S. Karni, J. J. Quirk, P. L. Roe, L. G. Simmonds, and E. F. Toro, *Numerical computation of two-dimensional unsteady detonation waves in high energy solids*, J. Comput. Phys., 106 (1993), pp. 215–233.

[6] P. Colella, H. M. Glaz, and R. E. Ferguson, *Multifluid algorithms for Eulerian finite difference methods*, unpublished manuscript, 1989 (revised, 1993).

[7] P. Colella, A. Majda, and V. Roytburd, *Theoretical and numerical structure for reacting shock waves*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 1059–1080.

[8] R. Courant and K. O. Friedrichs, *Supersonic flow and shock waves*, Interscience, New York, 1948.

[9] S. K. Godunov, *A difference method for the numerical calculation of discontinuous solutions of hydrodynamic equations*, Math. USSR-Sb., 47 (1959), p. 271.

[10] L. F. Henderson, P. Colella, and E. G. Puckett, *On the refraction of shock waves at a slow-fast gas interface*, J. Fluid Mech., 224 (1991), pp. 1–27.

[11] S. Karni, *Viscous shock profiles and primitive formulations*, SIAM J. Numer. Anal., 29 (1992), pp. 1592–1609.

[12] ———, *Multi-component flow calculations by a consistent primitive algorithm*, J. Comput. Phys., 112 (1994), pp. 31–43.

[13] B. Larrouturou, *How to preserve the mass fraction positive when computing compressible multi-component flows*, J. Comput. Phys., 95 (1991), pp. 59–84.

[14] B. Larrouturou and L. Fezoui, *On the equations of multicomponent perfect or real gas inviscid flow*, in Lecture Notes in Mathematics, vol. 1402, Springer-Verlag, Berlin, New York, 1989, pp. 69–97.

[15] R. J. LeVeque and H. C. Yee, *A study of numerical methods for hyperbolic conservation laws with stiff source terms*, J. Comput. Phys., 86 (1990), pp. 187–210.

[16] R. Menikoff, *Errors when shock waves interact due to numerical shock width*, SIAM J. Sci. Comput., 15 (1994), pp. 1227–1242.

[17] G. Moretti, *Thoughts and Afterthoughts About Shock Computations*, Technical Report PIBAL-72-37, Polytechnic Institute of New York, Brooklyn, New York, 1972.

[18] W. Mulder, S. Osher, and J. A. Sethian, *Computing interface motion: The compressible Rayleigh-Taylor and Kelvin-Helmholtz instabilities*, J. Comput. Phys., 100 (1992), p. 209.

[19] S. Osher and F. Solomon, *Upwind difference schemes for hyperbolic conservation laws*, Math. Comp., 38 (1982), pp. 339–374.

[20] J. E. Pilliod, Jr. and E. G. Puckett, *Second-order volume-of-fluid algorithms for tracking material interfaces*, manuscript.

[21] J. J. Quirk, and S. Karni, *On the dynamics of a shock-bubble interaction*, ICASE Technical Report 94-75. J. Fluid Mech., to appear.

[22] T. W. Roberts, *The behaviour of flux difference splitting schemes near slowly moving shock waves*, J. Comput. Phys., 90 (1990), pp. 141–160.

[23] P. L. Roe, *Approximate Riemann solvers, parameter vectors and difference schemes*, J. Comput. Phys., 43 (1981), pp. 357–372.

[24] ———, *Fluctuations and signals—A framework for numerical evolution problems*, in Numerical Methods for Fluid Dynamics, K. W. Morton and M. J. Baines, eds., Academic Press, New York, 1982, pp. 219–257.

[25] ———, *A New Approach to Computing Discontinuous Flows of Several Ideal Gases*, preprint, 1989.

[26] P. K. Sweby, *High resolution schemes using flux limiters for hyperbolic conservation laws*, SIAM J. Numer. Anal., 21 (1984), pp. 995–1011.

[27] V. T. Ton, A. R. Karagozian, B. E. Engquist, and S. J. Osher, *Numerical simulation of inviscid detonation waves with finite rate chemistry*, Paper 91-101, in Proc. Combustion Institute 1991 Fall Meeting, The University of California, Los Angeles, 1991.

[28] B. van Leer, *Towards the ultimate conservative difference scheme,* V: *A second order sequel to Godunov's method*, J. Comput. Phys., 32 (1979), pp. 101–136.

[29] G. Zwas and J. Roseman, *The effect of nonlinear transformations on computing weak solutions*, J. Comput. Phys., 12 (1973), pp. 179–186.

[30] G. H. Miller and E. G. Puckett, *Edge effects in molybdenum encapsulated molten silicate shock wave targets*, J. Appl. Phys., 75 (1994), pp. 1426–1434.

SIAM J. SCI. COMPUT.
Vol. 17, No. 5, pp. 1040–1067, September 1996

A PARALLEL IMPLEMENTATION OF THE *P*-VERSION OF THE FINITE ELEMENT METHOD*

YIMIN ZHU† AND I. NORMAN KATZ†

Abstract. An iterative method based on the textured decomposition (TD) is developed in order to solve the systems of linear equations arising in the p-version of the finite element method. The iteration is used to implement the p-version in parallel on an MIMD computer NCUBE/six. The objectives are twofold: to achieve high computational efficiency (that is, computational load should be balanced among the processors) and simultaneously to achieve rapid convergence.

A superelement, consisting of four adjacent rectangular finite elements, is constructed for two-dimensional problems. Based on the structural property of the shape functions, each superelement is partitioned into three blocks in two different ways, and a two-leaf TD is used. Computations for a superelement associated with each leaf are assigned to two processors and are performed in parallel. A new preconditioner is introduced to accelerate convergence in a preconditioned textured decomposition (PTD). A special local communication strategy is used to avoid global assembly and global communication.

Two model problems including a Laplace equation on a rectangular domain with a near singular solution and a Poisson equation on an L-shaped domain, are solved. The conjugate gradient (CG) method, the TD method, the recursive textured decomposition (RTD) method, both with and without preconditioning, and the classical iterative methods (Jacobi, Gauss–Seidel (GS), successive overrelaxation (SOR)), are used to solve both model problems. Load balance, speedup ratio, and spectral radii of the various iterations are studied. The test results indicate that recursive PTD with a local communication strategy gives at least a 30% improvement in computational time over the other methods.

Key words. p-version of the finite element method, parallel implementation

AMS subject classifications. 65N22, 65N30, 65F10

1. Introduction.

The finite element method is currently one of the most popular numerical methods used in the solution of engineering problems governed by partial differential equations. However, large problems, especially in three dimensions, are often not solvable on serial computers because of prohibitive computation time. Improvements in computing technology, based on large-scale parallel computers, offer the possibility of expanding the set of problems that can be solved efficiently. Such parallel computers appear attractive to finite element methods in which the local element provides a natural decomposition of the problem into (partially) computationally independent sets.

Variants of the finite element methods include the standard h-version, which uses low-order basis functions and achieves accuracy by refining meshes; the p-version, which uses a fixed mesh and achieves accuracy by increasing the order of the basis functions; and the h-p version, which combines these two approaches. For the first and last of these techniques, which divide the domain into local elements and compute associated local stiffness matrices, a large component of the required computations can be implemented very naturally on parallel computers [3]. In particular, for the h-version, domain decomposition methods [8], [9] group collections of elements into superelements. Construction of all local stiffness matrices associated with a superelement is independent of another superelement, so that they can be performed in parallel on separate processors. For the p- and h-p versions, we can think of the space of high-order basis functions in a single element as analogous to h-version grouping into superelements in the sense that construction of local operators can be done partially in parallel. The majority of elemental computations are performed on local data in a fully parallel manner.

*Received by the editors December 27, 1993; accepted for publication (in revised form) May 17, 1995. This paper is part of the doctoral research of author Yimin Zhu. The work of both authors was supported by the U.S. Air Force Office of Scientific Research under grant AFOSR 92-J-0043.

†Washington University, St. Louis, MO 63130 (yimin@hylas.wustl.edu, katz@zach.wustl.edu).

Direct methods are fully satisfactory for the solution of the resulting assembled system of linear equations for small-size problems or large sparse systems. But large-size problems (with more than 30,000 degrees of freedom [13]), which are common in the p-version of the finite element analysis, are inefficient both in terms of CPU time and storage. Iterative methods are a preferable choice for problems with a large number of elements and do not suffer from fill-in. Iterative methods can be implemented on parallel computers and achieve computational load balance; that is, the computational load can be balanced among processors. Classical iterative methods including Jacobi, Gauss–Seidel (GS), successive overrelaxation (SOR), symmetric successive overrelaxation (SSOR), and Chebyshev methods [7] have been studied thoroughly and solve well-conditioned systems efficiently. But these methods converge very slowly for the ill-conditioned systems. Multigrid methods have proved to be efficient solvers for problems arising by discretization of partial differential equations by the h-version finite element method [14], [15]. A straightforward application of multigridlike ideas to the p-version was studied by Brussino et al. [5], Foresti et al. [6], Mandel [13], [14], and Hu and Katz [10], [11]. The conjugate gradient (CG) method has been applied to solve discretized systems by the p-version in Babuska et al. [3], [4] in a parallel manner, and they have achieved impressive results. In this paper, a class of new parallel iterative algorithms based on a textured model decomposition is introduced to solve problems arising from discretization of elliptic partial differential equations by the p-version. Numerical results show that the textured decomposition (TD) methods converge faster than the CG method on sample problems. The recursive textured decomposition (RTD) method is also used in our scheme to achieve still better results. The model problems are intended only to compare the efficiency of iterative methods. For the relatively small number of unknowns in these problems (at most 2,500) direct methods could be more efficient. The comparisons given here for the various iterative methods are meant to provide a guide for their use in problems with large (more than 30,000) unknowns.

In recent years, preconditioners have been developed for the CG methods to accelerate the convergence. A preconditioner which uses the hierachic property of the p-version has been proposed by Babuska et al. [4]. It is shown in [4] that the condition number grows at most as $(\log p)^2$ for large p. We use a closely related preconditioner for the TD and RTD methods. The preconditioning successfully reduces the spectral radius of their iteration matrices.

This paper is organized as follows: in §2, we introduce the p-version of the finite element method [1]. We use the concepts from Szabo and Babuska's book [2] to give a brief introduction to one-dimensional and two-dimensional cases. The finite element space and the corresponding hierachic basis functions are introduced. Computations of elemental stiffness matrices and load vectors are discussed. Our finite element computations were made by using MSC/PROBE [17] and PEGASYS. We discuss assembly of the stiffness matrices and load vectors. Finally, we use the Schur complement to condense the internal variables and corresponding load vectors.

In §3, we construct the TD method. In order to apply the TD method to parallel computation, we introduce a superelement. A PTD method, based on structural properties of the superelement, is introduced. We study a three-level recursive TD with and without preconditioning.

In §4, we study parallel implementations of the TD method. First, we discuss how to map a superelement onto two processors. Then we propose a local communication strategy to minimize communication costs. Performance analysis focused on speedup ratios is studied.

In §5, we present the numerical results of using the TD method, the PTD method, and the RTD method to solve two model problems: a Laplace equation on a rectangular domain with near singular solution and an L-shaped domain problem. Both model problems are typical problems in elasticity for which finite element analysis is generally used. We compare these results with the results from using the classical iterative methods: Jacobi, GS, SOR, and the CG and preconditioned conjugate gradient (PCG) methods. Convergence rate and performance analysis is given.

2. The p-version of the finite element method. We consider the representative problem:

$$-\Delta u = f \quad \text{on } \Omega, \tag{2.1a}$$

$$u = f_1 \quad \text{on } \partial^1\Omega, \tag{2.1b}$$

$$\frac{\partial u}{\partial n} = f_2 \quad \text{on } \partial^2\Omega, \tag{2.1c}$$

where Ω is the bounded domain, $\partial^1\Omega \cup \partial^2\Omega = \partial\Omega$ is the piecewise smooth boundary, and f, f_1, f_2 are functions that satisfy the conditions guaranteeing existence and uniqueness of the solution.

Let

$$H(\Omega) \equiv \{u \in H^1(\Omega) | u = f_1 \text{ on } \partial^1\Omega\}, \tag{2.2}$$

where $H^1(\Omega)$ is a *Hilbert space.*

Let

$$B(u, v) = \int_\Omega (\nabla u \cdot \nabla v)\, dx_1 dx_2 \tag{2.3}$$

be the bilinear form defined on $H(\Omega) \times H(\Omega)$, and let

$$F(v) = \int_\Omega f v\, dx_1\, dx_2 + \int_{\partial^2\Omega} f_2 v\, ds \tag{2.4}$$

be a linear functional on $H(\Omega)$. Then we change problem (2.1) to the standard variational formulation: find

$$u_{EX} \in H(\Omega),$$

such that

$$B(u_{EX}, v) = F(v) \quad \forall v \in H(\Omega). \tag{2.5}$$

The p-version approximation consists of restricting the subspace $H(\Omega)$ and finding a solution in this restricted subspace. First, we choose a polynomial subspace $S^p \subseteq H(\Omega)$; then we find an approximate solution $u_{FE}(S^p) \in S^p$, such that

$$B(u_{FE}(S^p), v) = F(v) \quad \forall v \in S^p. \tag{2.6}$$

The goal is

$$\|u_{FE}(S^p) - u_{EX}\|_{E(\Omega)} \leq \tau, \tag{2.7}$$

where

$$\|u\|_{E(\Omega)} \equiv B(u, u)^{\frac{1}{2}} \tag{2.8}$$

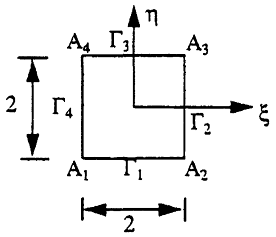


FIG. 2.1. *The standard quadrilateral element* $\Omega_{st}^{(q)}$.

is the *energy norm*, $E(\Omega)$ is the energy space defined as $E(\Omega) \equiv \{u|B(u,u) < \infty\}$, and τ is a given tolerance.

The finite element space in two dimensions is defined as follows:

$$S^p \equiv S^p(\Omega, \Delta, Q) \equiv \{\{u\}|\{u\} \in E(\Omega), u(Q_x^{(k)}(\xi,\eta), Q_y^{(k)}(\xi,\eta)) \in S_{st}^{p_k}, \quad k = 1, 2, \dots, M(\Delta)\}, \tag{2.9}$$

where Ω is the domain, Δ is the finite element mesh, and $x = Q_x^{(k)}(\xi,\eta)$, $y = Q_y^{(k)}(\xi,\eta)$, are mapping functions defined for the kth finite element which map a standard quadrilateral element onto the kth finite element. S_{st}^p is the space of polynomials defined on the standard elements [2].

In the finite element method, S^p is typically constructed by partitioning Ω into subdomains, called finite elements. Then the standard element $\Omega_{st}^{(q)}$ is selected. A standard square is shown in Fig. 2.1 which is mapped onto each finite element.

The space S_{st}^p is spanned by the following basis functions (for further details, see [2]):

1. *Nodal shape functions.* A nodal shape function is associated with a vertex A_i of the element $\Omega_{st}^{(q)}$. It is zero on the opposite sides of the vertex with which it is associated. There are four nodal shape functions:

$$N_1(\xi,\eta) = \frac{1}{4}(1-\xi)(1-\eta), \tag{2.10a}$$

$$N_2(\xi,\eta) = \frac{1}{4}(1+\xi)(1-\eta), \tag{2.10b}$$

$$N_3(\xi,\eta) = \frac{1}{4}(1+\xi)(1+\eta), \tag{2.10c}$$

$$N_4(\xi,\eta) = \frac{1}{4}(1-\xi)(1+\eta). \tag{2.10d}$$

2. *Side shape functions.* The side shape function is associated with a side Γ_i and is zero on all three other sides of the element. There are $4(p-1)$, $(p \geq 2)$ shape functions. These shape functions are defined as

$$N_i^{(1)} = \frac{1}{2}(1-\eta)\phi_i(\xi), \qquad i = 2, \dots, p, \tag{2.11a}$$

$$N_i^{(2)} = \frac{1}{2}(1+\xi)\phi_i(\eta), \qquad i = 2, \dots, p, \tag{2.11b}$$

$$N_i^{(3)} = \frac{(-1)^i}{2}(1+\eta)\phi_i(\xi), \qquad i = 2, \dots, p, \tag{2.11c}$$

$$N_i^{(4)} = \frac{(-1)^i}{2}(1-\xi)\phi_i(\eta), \qquad i = 2, \dots, p, \tag{2.11d}$$

where ϕ_j is defined in terms of the Legendre polynomial P_{j-1}:

$$\phi_j(\xi) \equiv \sqrt{\frac{2j-1}{2}} \int_{-1}^{\xi} P_{j-1}(t)\, dt, \qquad j = 2, 3, \ldots. \tag{2.12}$$

The term $(-1)^i$ is needed in $N_i^{(3)}$ and $N_i^{(4)}$ to obtain invariance with respect to the rotation of coordinates.

3. *Internal shape functions.* Internal shape functions are zero on all four sides. For $p < 4$ there are no internal shape functions. For $p \geq 4$ there are $(p-2)(p-3)/2$ internal shape functions defined as

$$N_{i,j}^{(0)}(\xi, \eta) = (1-\xi^2)(1-\eta^2)P_i(\xi)P_j(\eta), \qquad 0 \leq i + j \leq p - 4. \tag{2.13}$$

These basis functions are mapped onto each element. Let Ψ_i denote the mapped basis functions on an element; then we have

$$u_{FE}(S^p) = \sum_{i=1}^{N} x_i \Psi_i, \qquad F(\Psi_i) = y_i; \tag{2.14}$$

then problem (2.6) becomes the solution of a system of linear equations

$$Ax = y, \tag{2.15}$$

where

$A = [a_{ij}]$ is called the elemental *stiffness matrix* with $a_{ij} = B(\Psi_i, \Psi_j)$,
x is the vector of coefficients of basis functions,
y is called the elemental *load vector*.

After the elemental stiffness matrices are generated, we eliminate the internal unknowns by computing the Schur complement. The elemental stiffness matrices and load vectors can be written in block form:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{12}^T & A_{22} \end{bmatrix}, \qquad b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \tag{2.16}$$

where

A_{11} corresponds to the interior unknowns,
A_{22} corresponds to the boundary unknowns,
A_{12} corresponds to the connections between interior and boundary unknowns.

After computing the Schur complement, the stiffness matrices can be written in the following form:

$$\hat{A} = BAB^T = \begin{bmatrix} A_{11} & 0 \\ 0 & \hat{A}_{22} \end{bmatrix}, \tag{2.17}$$

where

$$\hat{A}_{22} = A_{22} - A_{12}^T A_{11}^{-1} A_{12}$$

is the Schur complement of A with respect to A_{11} and

$$B = \begin{bmatrix} I_1 & 0 \\ -A_{12}^T A_{11}^{-1} & I_2 \end{bmatrix}. \tag{2.18}$$

The elemental load vector will be modified similarly by

$$\hat{b} = [b_1, \hat{b}_2]^T, \tag{2.19a}$$

where

$$\hat{b}_2 = b_2 - A_{12}^T A_{11}^{-1} b_1. \tag{2.19b}$$

Remark. The condensation of internal unknowns is processed element by element, so these computations are fully parallizable.

DEFINITION 2.1. *Let* $A = (a_{ij})$ *be an* $n \times n$ *matrix with eigenvalues* λ_1, $1 \le i \le n$. *Then*

$$\rho(A) \equiv \max_{1 \le i \le n} |\lambda_i| \tag{2.20}$$

is the spectral radius *of the matrix* A.

3. Textured decomposition method.

3.1. The TD method. The TD method [12] for the p-version of the finite element method is a new class of linear iterative algorithms, using a class of parallel algorithms developed in different contexts at Washington University [18], [19], [20]. Unlike the standard iterative methods, the TD method uses multiple splittings of A called *leaves*, say (D_1, E_1), $(D_2, E_2), \ldots$, (D_m, E_m), and

$$A = D_1 - E_1 = D_2 - E_2 = \cdots = D_m - E_m. \tag{3.1}$$

The algorithm for the m-leaf TD method is defined as

$$\begin{aligned} &(3.2a) & D_1 x(k+1) &= E_1 x(k) + b, \\ &(3.2b) & D_2 x(k+2) &= E_2 x(k+1) + b, \\ && &\vdots \\ &(3.2c) & D_m x(k+m) &= E_m x(k+m-1) + b, \qquad k = 1, 2, \ldots. \end{aligned}$$

In each splitting (D_i, E_i), D_i is an approximation of A and $-E_i$ contains the remaining nonzeros of A. In the TD method, we use multiple approximations of A in round-robin fashion which yields a textured group of decompositions of the system matrix A. In this way, the approximation error incurred in one splitting is compensated by the other splittings in the overall TD. The central feature of our investigation is to find a choice of $D_1, E_1, \ldots, D_m, E_m$ for the p-version of the finite element method to assure rapid convergence and minimal idle time.

In our investigation, we focus on two-leaf TD. First we discuss how to choose the splittings. The system matrix A can be written in block form

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}; \tag{3.3}$$

then we choose the two splittings:

$$D_1 = \begin{bmatrix} A_{11} & 0 & 0 \\ 0 & A_{22} & A_{23} \\ 0 & A_{32} & A_{33} \end{bmatrix}, \qquad -E_1 = \begin{bmatrix} 0 & A_{12} & A_{13} \\ A_{21} & 0 & 0 \\ A_{31} & 0 & 0 \end{bmatrix}, \tag{3.4a}$$

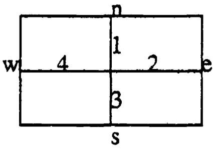


FIG. 3.1. *Superelement.*

$$D_2 = \begin{bmatrix} A_{11} & A_{12} & 0 \\ A_{21} & A_{22} & 0 \\ 0 & 0 & A_{33} \end{bmatrix}, \qquad -E_2 = \begin{bmatrix} 0 & 0 & A_{13} \\ 0 & 0 & A_{23} \\ A_{31} & A_{32} & 0 \end{bmatrix}. \tag{3.4b}$$

In the above decomposition, the dimensions of A_{11} and A_{33} are identical, say $m \times m$. Then the overlap between two leaves is of dimension $(n - 2m) \times (n - 2m)$. The two-leaf TD takes the form

$$D_1 x^{(k+1)} = E_1 x^{(k)} + b, \tag{3.5a}$$

$$D_2 x^{(k+2)} = E_2 x^{(k+1)} + b. \tag{3.5b}$$

Equations (3.5a,b) can be solved either by a direct solver or by an iterative solver. We discuss this later.

3.2. The TD method applied to the finite element method. In this paper, we consider the parallel implementations of the TD methods for solving the system of linear equations (3.1) derived from the discretization of a two-dimensional elliptic problem by the p-version of the finite element method on rectangular meshes. In order to be applicable to a parallel computer, first, we group the elements into clusters. We call each cluster a *superelement.* A superelement can be one element, two adjacent elements, or several adjacent elements. In our investigation, we use four adjacent quadrilateral elements as a superelement. (See Fig. 3.1.) Let A be the assembled stiffness matrix corresponding to a superelement. Our objectives are to partition A so as to achieve two goals: (1) rapid convergence and (2) maximum load balance in parallel computation.

Now we discuss how to partition the matrix A. Suppose matrix A is an assembled stiffness matrix for a superelement with all internal variables eliminated by computing the Schur complement. The matrix A can be reorganized as the block form

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}. \tag{3.6}$$

We consider two ways to partition these blocks, which we denote as BLOCK1 and BLOCK2. In BLOCK1, each block is chosen as follows:

A_{11} corresponds to all side variables of sides 1,2,n,e;

A_{22} corresponds to all vertex variables;

A_{33} corresponds to all side variables of sides 3,4,s,w.

In BLOCK2, each block is chosen as follows:

A_{11} corresponds to side variables of sides 1,2,n,e of order $\leq p - 1$;

A_{22} corresponds to all vertex variables and pth-order side variables;

A_{33} corresponds to side variables of sides 3,4,s,w of order $\leq p - 1$.

Remark. Block A_{22} is the overlap of the two leaves. It is a key part of our partitioning, which will help to accelerate the convergence. In our numerical investigations, we found

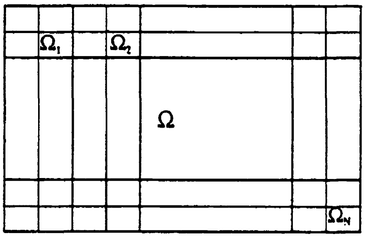


FIG. 3.2. *General case.*

that the error in the variables corresponding to vertices always decreased slower than other variables. If we put these vertex variables in the overlap, they will be computed twice (once in each leaf) per iteration. This will help to decrease the error. In BLOCK2, we use the hierachic property of the p-version of the finite element method and put the pth order of side variables in A_{22} as perturbation to the variables of order $\leq p-1$ to accelerate the convergence.

Generally, we use N superelements, each superelement contains four adjacent elements, and there are a total of $4N$ elements. (See Fig. 3.2.) The assembled stiffness matrix is in the form

$$A=\left[\begin{array}{c|c|c|c}
\begin{matrix}A_{11}&A_{12}&A_{13}\\A_{21}&A_{22}&A_{23}\\A_{31}&A_{32}&A_{33}\end{matrix} & X & X & X \\ \hline
X & \begin{matrix}A_{44}&A_{45}&A_{46}\\A_{54}&A_{55}&A_{56}\\A_{64}&A_{65}&A_{66}\end{matrix} & X & X \\ \hline
X & X & \ddots & X \\ \hline
X & X & X & \begin{matrix}A_{n-2,n-2}&A_{n-2,n-1}&A_{n-2,n}\\A_{n-1,n-2}&A_{n-1,n-1}&A_{n-1,n}\\A_{n,n-2}&A_{n,n-1}&A_{n,n}\end{matrix}
\end{array}\right]
\begin{array}{c}\Omega_1\\ \\ \Omega_2\\ \\ \\ \\ \Omega_N\end{array}.$$

The matrices indicated with X's correspond to coupling between different superelements. The diagonal blocks whose submatrices are denoted by A_{ij} have been emphasized because they are used in the textured decomposition.

Using the two-leaf TD method, D_i is chosen as

$$D_1=\left[\begin{array}{c|c|c|c}
\begin{array}{c|cc}A_{11}&0&0\\\hline 0&A_{22}&A_{23}\\0&A_{32}&A_{33}\end{array} & Y & Y & Y \\ \hline
Y & \begin{array}{c|cc}A_{44}&0&0\\\hline 0&A_{55}&A_{56}\\0&A_{65}&A_{66}\end{array} & Y & Y \\ \hline
Y & Y & \ddots & Y \\ \hline
Y & Y & Y & \begin{array}{c|cc}A_{n-2,n-2}&0&0\\\hline 0&A_{n-1,n-1}&A_{n-1,n}\\0&A_{n,n-1}&A_{n,n}\end{array}
\end{array}\right]
\begin{array}{c}\Omega_1\\ \\ \Omega_2\\ \\ \\ \\ \Omega_N\end{array}.$$

$$D_2=\left[\begin{array}{c|c|c|c}
\begin{array}{cc|c} A_{11} & A_{12} & 0 \\ A_{21} & A_{22} & 0 \\ \hline 0 & 0 & A_{33} \end{array} & Z & Z & Z \\
\hline
Z & \begin{array}{cc|c} A_{44} & A_{45} & 0 \\ A_{54} & A_{55} & 0 \\ \hline 0 & 0 & A_{66} \end{array} & Z & Z \\
\hline
Z & Z & \ddots & Z \\
\hline
Z & Z & Z & \begin{array}{cc|c} A_{n-2,n-2} & A_{n-2,n-1} & 0 \\ A_{n-1,n-2} & A_{n-1,n-1} & 0 \\ \hline 0 & 0 & A_{n,n} \end{array}
\end{array}\right]\begin{array}{c} \Omega_1 \\ \\ \Omega_2 \\ \\ \\ \Omega_N \end{array},$$

where $n = 3N$.

In the above splitting, each diagonal block corresponds to a superelement, so for N superelements, there are N diagonal blocks. Each block is defined as before. To solve equation (3.5a), we need to solve

$$A_{11}x_1^{(k+1)} = -\sum_{i\neq 1}^{n} A_{1i}x_i^{(k)} + b_1, \tag{3.7a}$$

$$\begin{bmatrix} A_{22} & A_{23} \\ A_{32} & A_{33} \end{bmatrix}\begin{bmatrix} x_2 \\ x_3 \end{bmatrix}^{(k+1)} = -\sum_{i\neq 2,3}^{n}\begin{bmatrix} A_{2i}x_i^{(k)} \\ A_{3i}x_i^{(k)} \end{bmatrix} + \begin{bmatrix} b_2 \\ b_3 \end{bmatrix}. \tag{3.7b}$$

For (3.5b), we need to solve

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^{(k+2)} = -\sum_{j\neq 1,2}^{n}\begin{bmatrix} A_{1j}x_j^{(k+1)} \\ A_{2j}x_j^{(k+1)} \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \tag{3.8a}$$

$$A_{33}x_3^{(k+2)} = -\sum_{j\neq 3}^{n} A_{3j}x_j^{(k+1)} + b_3. \tag{3.8b}$$

In (3.7a,b) and (3.8a,b), the system matrices, i.e., A_{11},

$$\begin{bmatrix} A_{22} & A_{23} \\ A_{32} & A_{33} \end{bmatrix}, \quad \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix},$$

and A_{33}, are all symmetric positive definite by the property of the p-version of the finite element discretization. These equations are solved using Cholesky factorization and a block forward solver.

From (3.6),

$$\begin{aligned} x^{(k+2)} &= D_2^{-1}E_2x^{(k+1)} + D_2^{-1}b \\ &= D_2^{-1}E_2D_1^{-1}E_1x^{(k)} + R, \end{aligned} \tag{3.9}$$

where

$$R = D_2^{-1}E_2D_1^{-1}b + D_2^{-1}b. \tag{3.10}$$

So the iteration matrix for the two-leaf TD method is

$$G_1 = D_2^{-1}E_2D_1^{-1}E_1, \tag{3.11}$$

and the convergence rate of this algorithm depends on the spectral radius of the iteration matrix G_1.

3.3. Preconditioning. We note that, for the iterative method, the system $M^{-1}Ax = M^{-1}b$ is called the *preconditioned system* of system (3.1). The matrix M is called the "preconditioner." In this section, we discuss the PCG method [7], the PTD method, and how to choose a proper preconditioner to accelerate the convergence.

3.3.1. The PTD method. In order to reduce the spectral radius of the iteration matrix G_1, we introduce a PTD method with preconditioner M^{-1}. The idea of a PTD method is to modify the residual in each iteration, then update x. The algorithm is as follows.

ALGORITHM 3.1 (PRECONDITIONED TEXTURED DECOMPOSITION METHOD).
Given a symmetric positive definite matrix $A \in \Re^{n\times n}$ and $b \in \Re^{n}$ we use a two-leaf TD method with preconditioner M and splittings D_1, D_2, E_1, and E_2 to solve the system of the linear equation $Ax = b$.

```
         Set $k = 0, x^0 = 0, r^0 = b$
(3.12a)  while $(\|x^{k+2} - x^k\| < \varepsilon)$
(3.12b)  Solve: $M^{-1}\tilde{r}^k = r^k$
(3.12c)  Update: $\tilde{x}^k = x^k + \tilde{r}^k$
(3.12d)  $D_1x^{k+1} = E_1\tilde{x}^k + b$
(3.12e)  $D_2x^{k+2} = E_2x^{k+1} + b$
         Update $k$
(3.12f)  $r^k = b - Ax^k$
End
```

Since the auxiliary system $M^{-1}\tilde{r} = r$ must be solved at each iteration in each preconditioned method, it is critical that this system be "easy" to solve. On the other hand, in order that the preconditioning be effective, we want preconditioner M to be a "good" approximation of A. Clearly, the two requirements are in conflict since the more closely M approximates A, the more likely that the system $M^{-1}\tilde{r} = r$ will be nearly as difficult to solve as the system $Ax = b$. We next discuss how to choose an efficient preconditioner so that the convergence is very fast and the equation $M^{-1}\tilde{r} = r$ is easy to solve.

3.3.2. Preconditioner. In the PTD method, from Algorithm 3.1,

$$\begin{aligned}
x^{k+2} &= G_1\tilde{x}^k + R && \text{(3.13a)}\\
&= G_1(x^k + \tilde{r}^k) + R && \text{(3.13b)}\\
&= G_1(x^k + Mr^k) + R && \text{(3.13c)}\\
&= G_1(x^k + M(b - Ax^k)) + R && \text{(3.13d)}\\
&= G_1(I - MA)x^k + G_1Mb + R, && \text{(3.13e)}
\end{aligned}$$

where G_1 and R are defined in (3.11) and (3.10), respectively.

The convergence rate of the PTD matrix depends on the spectral radius of the matrix

$$\tilde{G}_1 \equiv G_1(I - MA). \tag{3.14}$$

If $M^{-1} = A$, then $I - MA = 0$, and $\tilde{G}_1 = 0$; the iteration will terminate immediately.

In our investigation, we focus on choosing a proper preconditioner M^{-1} for the p-version approximation system equation, so that the spectral radius of iteration matrix $\tilde{G}_1$ is as small as possible.

More specifically consider a system of linear equations

$$Ax = b, \tag{3.15}$$

where A is the assembled stiffness matrix corresponding to a superelement with internal variables eliminated. We reform matrix A into block forms BLOCK1 and BLOCK2 as in §3.2. The preconditioner M will be chosen as

$$M = \begin{bmatrix} I & 0 & 0 \\ 0 & A_{22} & 0 \\ 0 & 0 & I \end{bmatrix}. \tag{3.16}$$

In §5, we show numerically that M^{-1} accelerates the convergence of the TD method. There are other preconditioners that have been used for the PCG method, such as the incomplete Cholesky factorization, polynomial preconditioners (SSOR, SOR, GS, etc.) [7], and the multigrid method as a preconditioner [15]. In §5, we study the use of the preconditioner M^{-1} defined in (3.16) for the PCG and the PTD methods.

3.4. The RTD method. In the p-version of the finite element method, we use polynomials of high degree for approximation. For larger p, each block of the matrix A is relatively big and load imbalance is considerable. So we divide each block of matrices into smaller blocks recursively and use the TD on the smaller blocks.

Consider the system of the linear equation

$$Ax = b, \tag{3.17}$$

where A is defined in §3.2 as BLOCK1.

We use a two-leaf TD to solve (3.17).

(i) *Level one.*

$$A = D_1 - E_1 = D_2 - E_2, \tag{3.18}$$

where

$$D_1 = \left[\begin{array}{c|cc} A_{11} & 0 & 0 \\ \hline 0 & A_{22} & A_{23} \\ 0 & A_{32} & A_{33} \end{array}\right], \qquad D_2 = \left[\begin{array}{cc|c} A_{11} & A_{12} & 0 \\ A_{21} & A_{22} & 0 \\ \hline 0 & 0 & A_{33} \end{array}\right], \tag{3.19}$$

are *leaves*. Let

$$D_1^1 \equiv [A_{11}], \qquad D_1^2 \equiv \begin{bmatrix} A_{22} & A_{23} \\ A_{32} & A_{33} \end{bmatrix}, \tag{3.20a}$$

$$D_2^1 \equiv \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \qquad D_2^2 \equiv [A_{33}], \tag{3.20b}$$

which we call *partition of leaves*

$$D_1 = \left[\begin{array}{c|c} D_1^1 & 0 \\ \hline 0 & D_1^2 \end{array}\right], \qquad D_2 = \left[\begin{array}{c|c} D_2^1 & 0 \\ \hline 0 & D_2^2 \end{array}\right]. \tag{3.21}$$

Similarly, we define E_1^1, E_1^2, E_2^1, and E_2^2.

At the first level, we solve

$$D_1 x^{k+1} = E_1 x^k + b, \tag{3.22a}$$

$$D_2 x^{k+2} = E_2 x^{k+1} + b, \tag{3.22b}$$

where

$$\begin{bmatrix} D_1^1 & 0 \\ 0 & D_1^2 \end{bmatrix} \begin{bmatrix} x_{1,1} \\ x_{1,2} \end{bmatrix}^{k+1} = \begin{bmatrix} 0 & E_1^1 \\ E_1^2 & 0 \end{bmatrix} \begin{bmatrix} x_{1,1} \\ x_{1,2} \end{bmatrix}^{k} + \begin{bmatrix} b_1^1 \\ b_1^2 \end{bmatrix}, \tag{3.23a}$$

$$\begin{bmatrix} D_2^1 & 0 \\ 0 & D_2^2 \end{bmatrix} \begin{bmatrix} x_{2,1} \\ x_{2,2} \end{bmatrix}^{k+2} = \begin{bmatrix} 0 & E_2^1 \\ E_2^2 & 0 \end{bmatrix} \begin{bmatrix} x_{2,1} \\ x_{2,2} \end{bmatrix}^{k+1} + \begin{bmatrix} b_2^1 \\ b_2^2 \end{bmatrix}; \tag{3.23b}$$

i.e., we solve

$$D_1^1 x_{1,1}^{k+1} = E_1^1 x_{1,2}^k + b_1^1, \tag{3.24a}$$

$$D_1^2 x_{1,2}^{k+1} = E_1^2 x_{1,1}^k + b_1^2, \tag{3.24b}$$

$$D_2^1 x_{2,1}^{k+2} = E_2^1 x_{2,2}^{k+1} + b_2^1, \tag{3.24c}$$

$$D_2^2 x_{2,2}^{k+2} = E_2^2 x_{2,1}^{k+1} + b_2^2. \tag{3.24d}$$

Note. Equations (3.24a) and (3.24b) can be solved simultaneously, (3.24c) and (3.24d) can be solved simultaneously, but (3.24c) and (3.24d) need results from (3.24a) and (3.24b). That is, the partitions of leaves can be solved in parallel, but leaves have to be solved sequentially.

(ii) *Level two.* To solve equations (3.24a–d), we use the two-leaf TD method again. First, we discuss how to solve (3.23a). We reorganize D_1^1 as follows:

$$D_1^1 = \begin{bmatrix} A_{11}^1 & A_{12}^1 & A_{13}^1 \\ A_{21}^1 & A_{22}^1 & A_{23}^1 \\ A_{31}^1 & A_{32}^1 & A_{33}^1 \end{bmatrix}, \tag{3.25}$$

where

A_{11}^1 corresponds to all side variables of side 1 and n,
A_{22}^1 corresponds to all vertex variables,
A_{33}^1 corresponds to all side variables of side 2 and e.

Let

$$D_1^1 = D_1^{1,1} - E_1^{1,1} = D_1^{1,2} - E_1^{1,2}, \tag{3.26}$$

where

$$D_1^{1,1} \equiv \left[\begin{array}{c|c} D_1^{1,1,1} & 0 \\ \hline 0 & D_1^{1,1,2} \end{array}\right] = \left[\begin{array}{c|cc} A_{11}^1 & 0 & 0 \\ \hline 0 & A_{22}^1 & A_{23}^1 \\ 0 & A_{32}^1 & A_{33}^1 \end{array}\right], \tag{3.27a}$$

$$D_1^{1,2} \equiv \left[\begin{array}{c|c} D_1^{1,2,1} & 0 \\ \hline 0 & D_1^{1,2,2} \end{array}\right] = \left[\begin{array}{cc|c} A_{11}^1 & A_{12}^1 & 0 \\ A_{21}^1 & A_{22}^1 & 0 \\ \hline 0 & 0 & A_{33}^1 \end{array}\right]. \tag{3.27b}$$

Now we need to solve

$$D_1^{1,1}x_{1,1}^{k_1+1} = E_1^{1,1}x_{1,2}^{k_1}+b_1^1, \tag{3.28a}$$

$$D_1^{1,2}x_{1,1}^{k_1+2} = E_1^{1,2}x_{1,2}^{k_1+1}+b_1^1, \tag{3.28b}$$

where k_1 is the iteration index in this level. We have

$$\begin{bmatrix} D_1^{1,1,1} & 0 \\ 0 & D_1^{1,1,2} \end{bmatrix}\begin{bmatrix} x_{1,1,1,1} \\ x_{1,1,1,2} \end{bmatrix}^{k_1+1} = \begin{bmatrix} 0 & E_1^{1,1,1} \\ E_1^{1,1,2} & 0 \end{bmatrix}\begin{bmatrix} x_{1,1,1,1} \\ x_{1,1,1,2} \end{bmatrix}^{k_1} + \begin{bmatrix} b_{1,1}^{1,1} \\ b_{1,1}^{1,2} \end{bmatrix}, \tag{3.29a}$$

$$\begin{bmatrix} D_1^{1,2,1} & 0 \\ 0 & D_1^{1,2,2} \end{bmatrix}\begin{bmatrix} x_{1,1,2,1} \\ x_{1,1,2,2} \end{bmatrix}^{k_1+2} = \begin{bmatrix} 0 & E_1^{1,2,1} \\ E_1^{1,2,2} & 0 \end{bmatrix}\begin{bmatrix} x_{1,1,2,1} \\ x_{1,1,2,2} \end{bmatrix}^{k_1+1} + \begin{bmatrix} b_{1,2}^{1,1} \\ b_{1,2}^{1,2} \end{bmatrix}; \tag{3.29b}$$

i.e., we solve

$$D_1^{1,1,1}x_{1,1,1,1}^{k_1+1} = E_1^{1,1,1}x_{1,1,1,2}^{k_1}+b_{1,1}^{1,1}, \tag{3.30a}$$

$$D_1^{1,1,2}x_{1,1,1,2}^{k_1+1} = E_1^{1,1,2}x_{1,1,1,1}^{k_1}+b_{1,1}^{1,2}, \tag{3.30b}$$

$$D_1^{1,2,1}x_{1,1,2,1}^{k_1+2} = E_1^{1,2,1}x_{1,1,2,2}^{k_1+1}+b_{1,2}^{1,1}, \tag{3.30c}$$

$$D_1^{1,2,2}x_{1,1,2,2}^{k_1+1} = E_1^{1,2,2}x_{1,1,2,1}^{k_1+1}+b_{1,2}^{1,2}. \tag{3.30d}$$

Similarly, we can use a two-leaf TD method to solve (3.24 b,c,d).

(iii) *Level three.* To solve equations (3.30a–d) in level two, we use two-leaf textured decomposition again. For equation (3.30a), we reorganize the block of matrix $D_1^{1,1,1}$ into the following form:

$$D_1^{1,1,1} = \begin{bmatrix} A_{11}^{1,1} & A_{12}^{1,1} & A_{13}^{1,1} \\ A_{21}^{1,1} & A_{22}^{1,1} & A_{23}^{1,1} \\ A_{31}^{1,1} & A_{32}^{1,1} & A_{33}^{1,1} \end{bmatrix}, \tag{3.31}$$

where

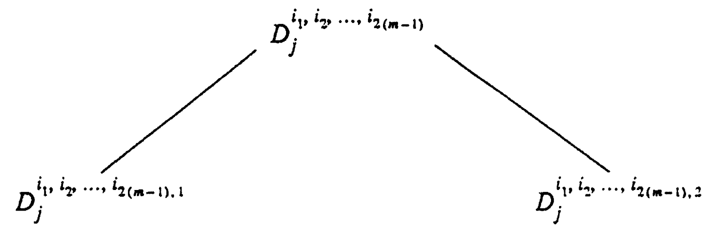
$A_{11}^{1,1}$ corresponds to side variables of side 1,
$A_{22}^{1,1}$ corresponds to all vertex variables,
$A_{33}^{1,1}$ corresponds to side variables of side n.

We then proceed as in BLOCK1 and BLOCK2.

A binary tree can be used to represent the RTD method. A *binary tree* is a tree where each node in the tree has 0, 1, or 2 children. Children are often referred to as "left child" or the "right child." The connection between children and their root parents are called *links*. We define the left link and the right link with respect to the connections between a parent with a left child and a right child. Figure 3.3 shows the mth-leaf binary tree. Figure 3.4 shows the three levels of the RTD. When we use preconditioning for the RTD method, we modify residuals in the final level by the preconditioner

$$M_j^{i_1,i_2,\ldots,i_{2(m-1)}} \equiv \begin{bmatrix} I & 0 & 0 \\ 0 & A_{22}^{l,k} & 0 \\ 0 & 0 & I \end{bmatrix}, \tag{3.32}$$

where $l, k = 1, 2$.



$D_j^{i_1, i_2, \ldots, i_{2(m-1)}}$
$D_j^{i_1, i_2, \ldots, i_{2(m-1)}, 1}$
$D_j^{i_1, i_2, \ldots, i_{2(m-1)}, 2}$

where $i_k, j = 1, 2$ and $k = 1, 2, \ldots, 2(m-1)$

FIG. 3.3. *mth-leaf binary tree.*

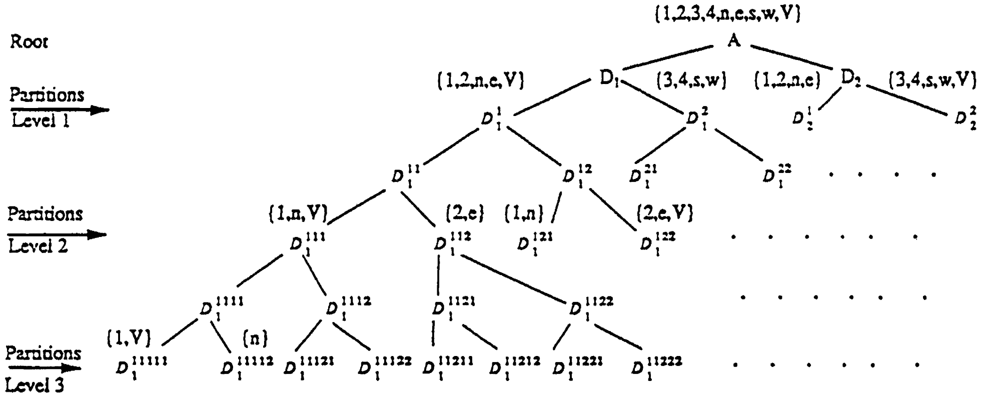


FIG. 3.4. *Three-level RTD.*

4. Parallel implementations. In this chapter, we discuss how to apply our iterative methods to a distributed parallel computer. We will discuss the mapping of a superelement onto parallel processors, communication strategy, and performance analysis. Our computations were performed on an NCUBE/6, a 64-processor MIMD parallel computer with hypercube architecture.

4.1. Superelements mapping. The hypercube interconnection system was chosen for three main reasons:

(i) It is an inductive structure; i.e., each node has its own memory, so it is usually easy to write programs that are independent of the hypercube.

(ii) It maps directly onto the most important common interconnection patterns.

(iii) It is so extensively interconnected that it gives a good approximation of maximal interconnection (every processor connected to all others). Thus, if a problem does not have an obvious interconnection structure, it will normally be acceptable to assign subproblems arbitrarily to hypercube nodes and let the communication software take care of routine messages [16].

In §3.2, we discussed how to construct a superelement. Now we map superelements onto a parallel computer. In our investigation, we use four adjacent elements as a superelement. We use a two-leaf TD method for each superelement and map each superelement onto two processors. Now we show how to map a superelement onto two processors. Suppose A is the global stiffness matrix defined in §3.2 for the general case

$$A=\left[\begin{array}{c|c|c|c}
\begin{matrix}A_{11}&A_{12}&A_{13}\\A_{21}&A_{22}&A_{23}\\A_{31}&A_{32}&A_{33}\end{matrix} & \mathrm{X} & \mathrm{X} & \mathrm{X} \\ \hline
\mathrm{X} & \begin{matrix}A_{44}&A_{45}&A_{46}\\A_{54}&A_{55}&A_{56}\\A_{64}&A_{65}&A_{66}\end{matrix} & \mathrm{X} & \mathrm{X} \\ \hline
\mathrm{X} & \mathrm{X} & \ddots & \mathrm{X} \\ \hline
\mathrm{X} & \mathrm{X} & \mathrm{X} & \begin{matrix}A_{n-2,n-2}&A_{n-2,n-1}&A_{n-2,n}\\A_{n-1,n-2}&A_{n-1,n-1}&A_{n-1,n}\\A_{n,n-2}&A_{n,n-1}&A_{n,n}\end{matrix}
\end{array}\right]
\begin{array}{c}\Omega_1\\ \\ \Omega_2\\ \\ \\ \\ \Omega_N\end{array}.$$

When using the two-leaf TD method, we solve the following equations in each superelement. In the first-leaf method we solve

$$A_{11}x_1^{(k+1)} = -\sum_{i\neq 1}^{n} A_{1i}x_i^{(k)} + b_1, \tag{4.1a}$$

$$\begin{bmatrix} A_{22} & A_{23} \\ A_{32} & A_{33} \end{bmatrix}\begin{bmatrix} x_2 \\ x_3 \end{bmatrix}^{(k+1)} = -\sum_{i\neq 2,3}^{n}\begin{bmatrix} A_{2i}x_i^{(k)} \\ A_{3i}x_i^{(k)} \end{bmatrix} + \begin{bmatrix} b_2 \\ b_3 \end{bmatrix}, \tag{4.1b}$$

and in the second-leaf method we solve

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^{(k+2)} = -\sum_{j\neq 1,2}^{n}\begin{bmatrix} A_{1j}x_j^{(k+1)} \\ A_{2j}x_j^{(k+1)} \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \tag{4.2a}$$

$$A_{33}x_3^{(k+2)} = -\sum_{j\neq 3}^{n} A_{3j}x_j^{(k+1)} + b_3. \tag{4.2b}$$

Equation (4.1a) will be solved in one processor and (4.1b) will be solved simultaneously in another processor. Similarly, we solve (4.2a) in the same processor where we solve (4.1a), and solve (4.2b) in the processor where we solve (4.1b). In this way, we map one superelement onto two processors.

4.2. Special communication strategy. In §3, we discussed how to construct and assemble the stiffness matrix and how to partition the assembled stiffness matrix into two kinds of blocks. Now we reconstruct and assemble the stiffness matrix so that we only need to assemble stiffness matrices locally (superelementwise) and communicate between neighbors. This is important on an N cube because communication between neighboring processors is much faster than between distant processors.

Let A be the assembled stiffness matrix for a superelement Ω_e with internal variables eliminated. See Fig. 4.1. A can be written as

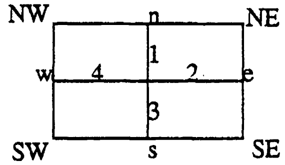


FIG. 4.1. *Superelement* Ω_e.

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}. \tag{4.3}$$

In §3.2, we have discussed two ways of partitioning A.

For BLOCK1, each block of matrix A is chosen such that

A_{11} corresponds to all side variables of sides 1,2,n,e;

A_{22} corresponds to all vertex variables;

A_{33} corresponds to all side variables of sides 3,4,s,w.

For BLOCK2, each block of matrix A is chosen such that

A_{11} corresponds to side variables of sides 1,2,n,e of order $\leq p-1$;

A_{22} corresponds to all vertex variables and pth order side variables;

A_{33} corresponds to side variables of sides 3,4,s,w of order $\leq p-1$.

$$A = \left[\begin{array}{c|c|c} A_{11} & A_{12} & A_{13} \\ \hline A_{21} & A_{22} & A_{23} \\ \hline A_{31} & A_{32} & A_{33} \end{array}\right] = \left[\begin{array}{c|cc|ccc} A'_{11} & A'_{12} & A'_{13} & A'_{14} & A'_{15} & A'_{16} \\ \hline A'_{21} & A'_{22} & A'_{23} & A'_{24} & A'_{25} & A'_{26} \\ A'_{31} & A'_{32} & A'_{33} & A'_{34} & A'_{35} & A'_{36} \\ \hline A'_{41} & A'_{42} & A'_{43} & A'_{44} & A'_{45} & A'_{46} \\ A'_{51} & A'_{52} & A'_{53} & A'_{54} & A'_{55} & A'_{56} \\ A'_{61} & A'_{62} & A'_{63} & A'_{64} & A'_{65} & A'_{66} \end{array}\right]. \tag{4.4}$$

Now we reconstruct A in the following way. Instead of assembling the global stiffness matrix, we assemble the stiffness matrix of each superelement. So, for each superelement, the assembled stiffness matrix takes the form BLOCK1, where

A'_{11} corresponds to all side variables of sides 1,2 and boundary n,e;

A'_{22} corresponds to all vertex variables excluding SW;

A'_{33} corresponds to vertex variables on SW;

A'_{44} corresponds to all side variables of sides 3 and 4;

A'_{55} corresponds to side variables of boundary w;

A'_{66} corresponds to side variables of boundary s.

From (4.4) we see that A'_{11} is the same as A_{11}, A_{22} is broken into two parts with diagonals A'_{22} and A'_{33}, and A_{33} is broken into three parts with diagonals A'_{44}, A'_{55}, and A'_{66}. Let $A'_{33}=0$, $A'_{55}=0$, and $A'_{66}=0$. We have the new stiffness matrix for the superelement

$$A' = \left[\begin{array}{c|cc|ccc} A'_{11} & A'_{12} & A'_{13} & A'_{14} & A'_{15} & A'_{16} \\ \hline A'_{21} & A'_{22} & A'_{23} & A'_{24} & A'_{25} & A'_{26} \\ A'_{31} & A'_{32} & 0 & A'_{34} & A'_{35} & A'_{36} \\ \hline A'_{41} & A'_{42} & A'_{43} & A'_{44} & A'_{45} & A'_{46} \\ A'_{51} & A'_{52} & A'_{53} & A'_{54} & 0 & A'_{56} \\ A'_{61} & A'_{62} & A'_{63} & A'_{64} & A'_{65} & 0 \end{array}\right]. \tag{4.5}$$

Let

$$A' = D_1 - E_1 = D_2 - E_2, \tag{4.6}$$

where

$$D_1 = \left[\begin{array}{ccc|ccc} A'_{11} & A'_{12} & A'_{13} & 0 & 0 & 0 \\ A'_{21} & A'_{22} & A'_{23} & 0 & 0 & 0 \\ A'_{31} & A'_{32} & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & A'_{44} & A'_{45} & A'_{46} \\ 0 & 0 & 0 & A'_{54} & 0 & A'_{56} \\ 0 & 0 & 0 & A'_{64} & A'_{65} & 0 \end{array}\right], \tag{4.7}$$

$$D_2 = \left[\begin{array}{c|ccccc} A'_{11} & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & A'_{22} & A'_{23} & A'_{24} & A'_{25} & A'_{26} \\ 0 & A'_{32} & 0 & A'_{34} & A'_{35} & A'_{36} \\ 0 & A'_{42} & A'_{43} & A'_{44} & A'_{45} & A'_{46} \\ 0 & A'_{52} & A'_{53} & A'_{54} & 0 & A'_{56} \\ 0 & A'_{62} & A'_{63} & A'_{64} & A'_{65} & 0 \end{array}\right]. \tag{4.8}$$

Similarly, we define E_1 and E_2.

Using the two-leaf TD method we have

$$D_1 x^{k+1} = E_1 x^k + b, \tag{4.9}$$

$$D_2 x^{k+2} = E_2 x^{k+1} + b, \tag{4.10}$$

where

$$x^k = [x_1^k, x_2^k, x_3^k, x_4^k, x_5^k, x_6^k]^T;$$

i.e.,

$$\left[\begin{array}{ccc|ccc} A'_{11} & A'_{12} & A'_{13} & 0 & 0 & 0 \\ A'_{21} & A'_{22} & A'_{23} & 0 & 0 & 0 \\ A'_{31} & A'_{32} & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & A'_{44} & A'_{45} & A'_{46} \\ 0 & 0 & 0 & A'_{54} & 0 & A'_{56} \\ 0 & 0 & 0 & A'_{64} & A'_{65} & 0 \end{array}\right] \left[\begin{array}{c} x_1 \\ x_2 \\ x_3 \\ \hline x_4 \\ x_5 \\ x_6 \end{array}\right]^{(k+1)}$$

$$= \left[\begin{array}{ccc|ccc} 0 & 0 & 0 & A'_{14} & A'_{15} & A'_{16} \\ 0 & 0 & 0 & A'_{24} & A'_{25} & A'_{26} \\ 0 & 0 & 0 & A'_{34} & A'_{35} & A'_{36} \\ \hline A'_{41} & A'_{42} & A'_{43} & 0 & 0 & 0 \\ A'_{51} & A'_{52} & A'_{53} & 0 & 0 & 0 \\ A'_{61} & A'_{62} & A'_{63} & 0 & 0 & 0 \end{array}\right] \left[\begin{array}{c} x_1 \\ x_2 \\ x_3 \\ \hline x_4 \\ x_5 \\ x_6 \end{array}\right]^{(k)} + b, \tag{4.11a}$$

$$\left[\begin{array}{c|ccccc} A'_{11} & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & A'_{22} & A'_{23} & A'_{24} & A'_{25} & A'_{26} \\ 0 & A'_{32} & 0 & A'_{34} & A'_{35} & A'_{36} \\ 0 & A'_{42} & A'_{43} & A'_{44} & A'_{45} & A'_{46} \\ 0 & A'_{52} & A'_{53} & A'_{54} & 0 & A'_{56} \\ 0 & A'_{62} & A'_{63} & A'_{64} & A'_{65} & 0 \end{array}\right] \left[\begin{array}{c} x_1 \\ \hline x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array}\right]^{(k+2)}$$

$$\text{(4.11b)} \qquad = \left[\begin{array}{c|ccccc} 0 & A'_{12} & A'_{13} & A'_{14} & A'_{15} & A'_{16} \\ \hline A'_{21} & 0 & 0 & 0 & 0 & 0 \\ A'_{31} & 0 & 0 & 0 & 0 & 0 \\ A'_{41} & 0 & 0 & 0 & 0 & 0 \\ A'_{51} & 0 & 0 & 0 & 0 & 0 \\ A'_{61} & 0 & 0 & 0 & 0 & 0 \end{array}\right] \left[\begin{array}{c} x_1 \\ \hline x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array}\right]^{(k+1)} + b.$$

In (4.11a,b), each matrix block is mapped onto a processor. After one iteration, we send x_5, which is associated with boundary w, to the neighbor processors on the west; send x_6, which is associated with boundary s, to the neighbors on the south; send x_3, which is associated with vertex SW, to the neighbors on the south, west, and southwest. Meanwhile, Ω_e receives values from the neighbors on the north, east, and northeast. (See Fig. 4.2.) Then we communicate values between the two processors associated with the same superelement, then update. In this way, we avoid assembly of the global stiffness matrix and global communication. Intuitively speaking, the west boundary w is a common boundary of superelement Ω_e and its superelement to the west. So the variables x_5, which are the variables associated with boundary w, depend on both superelement Ω_e and its western superelement. After each iteration, superelement Ω_e sends x_5 to its western superelement. The western superelement will add the received x_5 to its x_1 which is associated with the western node of Ω_e, i.e., w.

For BLOCK2, we reconstruct the local stiffness for a superelement as follows:

$$\text{(4.12)} \quad A = \left[\begin{array}{c|c|c} A_{11} & A_{12} & A_{13} \\ \hline A_{21} & A_{22} & A_{23} \\ \hline A_{31} & A_{32} & A_{33} \end{array}\right] = \left[\begin{array}{c|cccc|ccc} A'_{11} & A'_{12} & A'_{13} & A'_{14} & A'_{15} & A'_{16} & A'_{17} & A'_{18} \\ \hline A'_{21} & A'_{22} & A'_{23} & A'_{24} & A'_{25} & A'_{26} & A'_{27} & A'_{28} \\ A'_{31} & A'_{32} & A'_{33} & A'_{34} & A'_{35} & A'_{36} & A'_{37} & A'_{38} \\ A'_{41} & A'_{42} & A'_{43} & A'_{44} & A'_{45} & A'_{46} & A'_{47} & A'_{48} \\ A'_{51} & A'_{52} & A'_{53} & A'_{54} & A'_{55} & A'_{56} & A'_{57} & A'_{58} \\ \hline A'_{61} & A'_{62} & A'_{63} & A'_{64} & A'_{65} & A'_{66} & A'_{67} & A'_{68} \\ A'_{71} & A'_{72} & A'_{73} & A'_{74} & A'_{75} & A'_{76} & A'_{77} & A'_{78} \\ A'_{81} & A'_{82} & A'_{83} & A'_{84} & A'_{85} & A'_{86} & A'_{87} & A'_{88} \end{array}\right],$$

where

A'_{11} corresponds to all side variables of sides 1,2, n,e of order $\leq p - 1$;

A'_{22} corresponds to all vertex variables excluding SW, and the pth-order side variables of sides 1, 2, 3, 4, n, and e;

A'_{33} corresponds to vertex variables on SW;

A'_{44} corresponds to pth-order side variables on side w;

A'_{55} corresponds to pth-order side variables on side s;

A'_{66} corresponds to all side variables on 3,4 of order $\leq p - 1$;

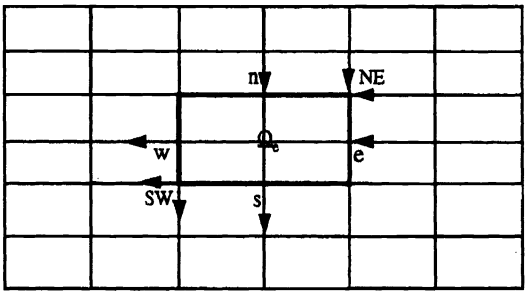


FIG. 4.2. *Local communication.*

A'_{77} corresponds to all side variables on w of order $\leq p-1$;
A'_{88} corresponds to all side variables on s of order $\leq p-1$.
Now we let $A'_{33}=0$, $A'_{44}=A'_{55}=0$, $A'_{77}=A'_{88}=0$; stiffness matrix A is changed to

$$A' = \left[\begin{array}{c|cccc|ccc} A'_{11} & A'_{12} & A'_{13} & A'_{14} & A'_{15} & A'_{16} & A'_{17} & A'_{18} \\ \hline A'_{21} & A'_{22} & A'_{23} & A'_{24} & A'_{25} & A'_{26} & A'_{27} & A'_{28} \\ A'_{31} & A'_{32} & 0 & A'_{34} & A'_{35} & A'_{36} & A'_{37} & A'_{38} \\ A'_{41} & A'_{42} & A'_{43} & 0 & A'_{45} & A'_{46} & A'_{47} & A'_{48} \\ A'_{51} & A'_{52} & A'_{53} & A'_{54} & 0 & A'_{56} & A'_{57} & A'_{58} \\ \hline A'_{61} & A'_{62} & A'_{63} & A'_{64} & A'_{65} & A'_{66} & A'_{67} & A'_{68} \\ A'_{71} & A'_{72} & A'_{73} & A'_{74} & A'_{75} & A'_{76} & 0 & A'_{78} \\ A'_{81} & A'_{82} & A'_{83} & A'_{84} & A'_{85} & A'_{86} & A'_{87} & 0 \end{array}\right]. \tag{4.13}$$

Similarly, we define D_1, D_2, E_1, E_2; then using two-leaf TD method we have

$$D_1 x^{k+1} = E_1 x^k + b, \tag{4.14}$$

$$D_2 x^{k+2} = E_2 x^{k+1} + b, \tag{4.15}$$

where

$$x^k = [x_1^k, x_2^k, x_3^k, x_4^k, x_5^k, x_6^k, x_7^k, x_8^k]^T.$$

4.3. Performance analysis. Our performance metric for a parallel computer is the *speedup* factor S associated with a particular calculation. The way *speedup* is defined varies widely. The following are two common formulations for speedup:

$$S_N(p) = \frac{T_{seq}(p)}{T_N(p)}, \tag{4.16}$$

$$S_N(p) = \frac{T_1(p)}{T_N(p)}, \tag{4.17}$$

where $T_N(p)$ is the time required to solve a problem in parallel on N processors, $T_1(p)$ is the time required to execute a parallel program on one processor, while T_{seq} is the time required to execute an equivalent sequential program on one processor.

It is unrealistic to use equation (4.17), since $T_1(p)$ includes all the overhead of parallel computing. We choose equation (4.16) to calculate the speedup.

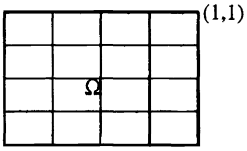


FIG. 5.1. *Model problem* 1.

Note. (1) Speedup is dependent on both the number of processors used and the size of the problem to be solved on the parallel computer. The polynomial degree p determines the size of the problem.

(2) The sequential computing time T_{seq} is measured by running the program (no parallel constructs at all) on a single processor. If T_{seq} cannot be measured directly, then it can usually be found by running a small problem on one processor and then extrapolating to the real problem by using a predicted or measured dependence on problem size.

5. Numerical results and comparisons. In previous sections, we discussed the p-version of the finite element method and how to use this method to discretize elliptic partial differential equations; we then introduced a new type of iterative method, the TD method, to solve the system of linear equations generated by the p-version of the finite element method on a parallel computer. To determine whether the TD method converges faster than the classical iterative methods and CG method, we use numerical experiments for solving two model problems. These experiments demonstrate how efficient the TD method is, compared with other classical iterative methods.

5.1. Model problem 1: Rectangular domain, near singular solution. Now we discuss a model problem, a Laplace equation on a rectangular domain with a near singular solution. Consider the equation

$$\Delta u = 0 \quad \text{on} \quad \Omega = [0,1] \times [0,1], \tag{5.1}$$

with

$$\frac{\partial u}{\partial n}(x,0) = \frac{\partial u}{\partial n}(0,y) = 0, \tag{5.2}$$

$$\frac{\partial u}{\partial n}(X,1) = g_1(x), \tag{5.3}$$

$$\frac{\partial u}{\partial n}(1,y) = g_2(y), \tag{5.4}$$

where $g_1(x)$ and $g_2(y)$ are determined so that the solution is

$$u(x,y) = \operatorname{Re}[(a^2+z^2)^{-1} + (a^2-z^2)^{-1}] - 2a, \tag{5.5}$$

with constant $a > 1$ and $z = x + iy$. This example is taken from Babuska et al. [4]. See Fig. 5.1.

Note that the solution u is a harmonic function with a singularity at $z = \pm a, \pm ia$. The solution becomes less smooth as $a \to 1$. This is a typical model problem in structural mechanics, where the solution has a singularity in a known location. The parameter a characterizes the regularity of the solution. We solve this problem for $a = 1.05$.

We divide the domain into four superelements (see Fig. 5.2); then we solve it with eight processors. Using the p-version approximation (the finite element computations were made

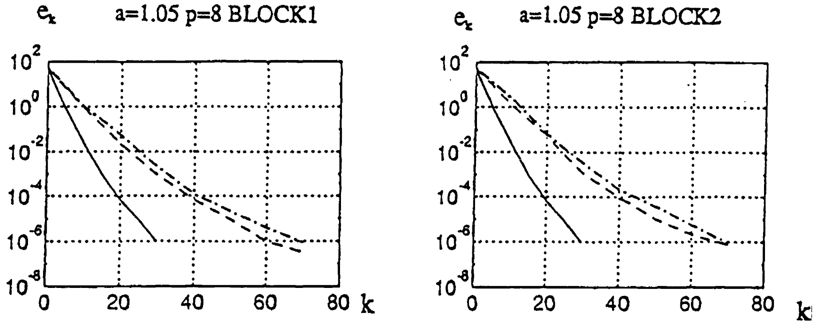


FIG. 5.2. *Convergence of iterative methods.*

TABLE 5.1
Comparison of number of iterations.

	RTD	TD	CG	J	GS	SOR
B1	36	62	66	251	161	133
B2	30	52	66	251	161	133

TABLE 5.2
Comparison of number of iterations (preconditioned iterations).

a	PCG(B1)	PTD(B1)	PRTD(B1)	PCG(B2)	PTD(B2)	PRTD(B2)
1.05	50	42	28	45	38	24

with the commercial code MSC/PROBE), we generate the elemental stiffness matrix and load vector. We eliminate the internal variables elementwise; then we assemble the stiffness matrix and the load vector. We solve the following system of the linear equation

$$Ax = b, \tag{5.6}$$

where A and b are the assembled stiffness matrix and load vector, respectively.

We choose initial value randomly between 0 and 1, and the iteration is stopped when $\|x^k - x\| \leq 10^{-6}$. The true solution x is achieved when the iteration satisfies $\|x^{k+2} - x^k\| \leq 10^{-12}$. First we compare the number of iterations of the RTD, two-leaf TD, CG, Jacobi (J), GS, and SOR methods for $p = 8$. Table 5.1 shows the results of the number of iterations for $p = 8$ using partition BLOCK1 and BLOCK2.

Remark 5.1. From Table 5.1 we see that both the RTD and the TD methods converge much faster than the classical iterative method and about 30% to 60% faster than the CG method. The RTD method converges 30% to 40% faster than the nonrecursive TD method.

Table 5.2 shows he results of the number of iterations of the PCG, PTD, and PRTD methods for different partitions. Compared with the results in Table 5.1, we see that the preconditioned methods converge faster than unpreconditioned methods. The PRTD method has the fastest convergence rate among these iterative methods. It seems that when we use more superelements, BLOCK2 will accelerate the convergence slightly compared with BLOCK1 but at the high cost of the communication and programming complexity. We suggest that for a small number of superelements, BLOCK2 should be used to achieve best performance; for a large number of superelements, BLOCK1 should be used.

Figure 5.2 presents the convergence of the iterative methods: the RTD method (solid line), the TD method (dashed line), and the CG method (dashed-dotted line) for different partitions.

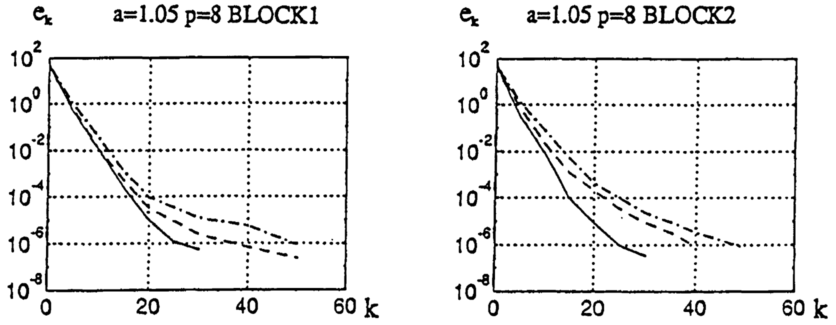


FIG. 5.3. *Convergence of iterative methods (preconditioned).*

TABLE 5.3
Comparison of spectral radius.

a	TD(B1)	PTD(B1)	RTD(B1)	PRTD(B1)	TD(B2)	PTD(B2)	RTD(B2)	PRTD(B2)
1.05	0.741	0.708	0.641	0.580	0.741	0.699	0.631	0.551

Figure 5.3 shows the convergence of the PRTD method (solid line), the PTD method (dashed line), and the PCG method (dashed-dotted line) for different partitions. The left figures of both Figs. 5.2 and 5.3 are BLOCK1 case, and right figures are BLOCK2 case. e_k is the error; k is the number of iterations. Comparing Fig. 5.3 with Fig. 5.2, it is clear that the preconditioning accelerates the convergence. RTD converges faster than TD and CG; PRTD converges faster than PTD and PCG. BLOCK2 improves the convergence, but communication is expensive. This will be shown in performance analysis.

Table 5.3 shows the spectral radius of the iteration matrices for TD, PTD, RTD, and PRTD for different partitions. For TD, the iteration matrix is $G_1 = D_2^{-1}E_2D_1^{-1}E_1$ (see equation (3.22)); for PTD, the iteration matrix is $\tilde{G}_1 \equiv G_1(I - M^{-1}A)$. (See equation (3.22).) When we use RTD and PRTD, we use equation $\rho(A) \approx \frac{e_{k+1}}{e_k}$ to compute the spectral radius.

Remark 5.2. Table 5.3 shows that the PRTD method has the smallest spectral radius among those iterative methods in the table. BLOCK2 improves the spectral radius a little but not significantly, but requires more time to communicate among each processors. The initial error e^0 increases because x^0 is chosen between 0 and 1, whereas the exact x is of larger magnitude. Therefore more iterations are needed to achieve the given accuracy.

In the performance analysis, first we present the execution time required for each iterative method for both BLOCK1 and BLOCK2. Both global communication and local communication strategies are used in our numerical experiment. The local communication strategy, as introduced in 4.3, is also used in the CG method. Then we implement the various algorithms on 2, 4, 8, 16, and 32 processors and compute the speedup ratios of these applications. In the following table, L refers to the local communication strategy, and G refers to the global communication strategy.

Tables 5.4 and 5.5 show the execution times of the TD, PTD, RTD, and PRTD methods running on 16 processors for different partitions for both global communication strategy and local communication strategy. The data shows that it takes more time to use BLOCK2 than to use BLOCK1. Local communication strategy has greatly reduced the communication time (more than 50%).

Figure 5.4 shows the speedup of the RTD method (solid line), the TD method (dashed-dotted line), and the CG method (dashed line) achievable for different choices of parameters and communication strategy. The left-hand side of the figure shows the speedup ratio using

TABLE 5.4
Comparison of time (sec.) (BLOCK1).

	TD(L)	PTD(L)	RTD(L)	PRTD(L)	TD(G)	PTD(G)	RTD(G)	PRTD(G)
CPU	103.35	79.32	73.05	55.01	112.91	88.27	82.19	61.92
Comm.	6.08	5.54	5.46	5.28	14.05	12.23	11.44	10.20
Comp.	89.82	70.02	59.54	43.18	89.95	70.11	60.04	43.97

TABLE 5.5
Comparison of time (sec.) (BLOCK2).

	TD(L)	PTD(L)	RTD(L)	PRTD(L)	TD(G)	PTD(G)	RTD(G)	PRTD(G)
CPU	112.44	87.91	78.74	61.25	128.11	100.01	91.35	66.59
Comm.	15.09	13.21	11.34	10.02	28.49	20.33	18.85	15.01
Comp.	86.21	67.58	57.04	41.22	86.20	67.45	57.09	41.14

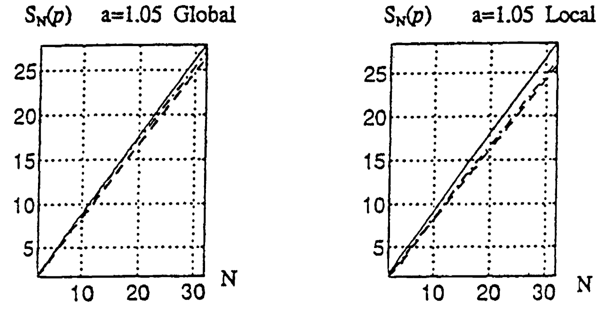


FIG. 5.4. *Speedup ratio.*

the global communication strategy, and the local communication strategy is shown on the right-hand side. It shows that the local communication strategy increases the speedup ratios. In most cases, the RTD method has the best performance.

From the results of model problem 1, it is clear that the PRTD method is the winner among the various iterative methods we discussed both on convergence rate and communication time. The preconditioner we used accelerates the convergence of the CG, TD, and RTD methods. The local communication strategy saves almost 50% of the communication time required for the global communication. BLOCK2 requires more communication than BLOCK1 for the multiple superelements case.

Similar results have been obtained for $p = 6$, $a = 1.5, 1.1, 1.05$, and for $p = 8$, $a = 1.5, 1.1$.

5.2. Model problem 2: L-shaped domain problem. The second model problem is a Poisson equation on an L-shaped domain. Consider the model problem, a two-dimensional Poisson equation

$$-\Delta u = f \quad \text{in } \Omega \tag{5.7}$$

with boundary conditions

$$\begin{aligned} &u = 0 \quad \text{on } \Gamma^2, \Gamma^3,\\ &\frac{\partial u}{\partial n} = 0 \quad \text{on } \Gamma^1, \Gamma^4,\\ &\frac{\partial u}{\partial n} = 1 \quad \text{on } \Gamma^5, \Gamma^6. \end{aligned}$$

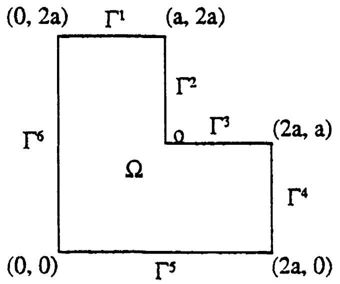


FIG. 5.5. *Model problem* 2.

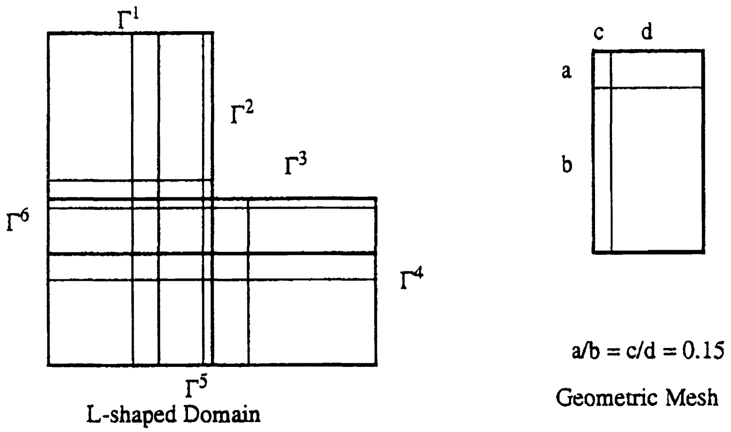


FIG. 5.6. *Mesh refinement.*

Assume $f = 0$; the exact solution is

$$u_0(r, \theta) = \sum_{k=1}^{\infty} a_k r^{(\frac{2}{3})k} \sin\left(\frac{2}{3}\right) k\theta. \tag{5.8}$$

From the exact solution we know that there is a singularity at point O. (See Fig. 5.5.) So we use a geometric mesh (see [2]) to divide the L-shaped domain into eight superelements which consist of 32 elements. (See Fig. 5.6.) We use the p-version finite element method discretization of the partial differential equation. The finite element computations were made with a test code called PEGASYS, which is under development by Engineering Software Research and Development, Inc. After we generate the elemental stiffness matrices and load vectors, we eliminate the internal unknowns, then assemble the elemental stiffness matrices into one global stiffness. We are now solving the system of linear equations

$$Ax = b, \tag{5.9}$$

where A is the assembled stiffness matrix, and b is the assembled load vector. To solve equation (5.9), we again use two kinds of communications: global communication, which communicates among all the processors, and local communication, which assembles stiffness matrices only in each superelement and communicates through its neighbors. Since the classical iterative methods, Jacobi, GS, and SOR, take more than 500 iterations (the limit we set)

TABLE 5.6
Comparison of the number of iterations.

	RTD	PRTD	TD	PTD	CG	PCG
$p=6$	72	58	104	78	112	98
$p=7$	74	64	106	80	115	101
$p=8$	82	76	112	84	126	105

TABLE 5.7
Comparison of the spectral radius for different iterations.

	RTD	PRTD	TD	PTD
$p=6$	0.815	0.771	0.875	0.842
$p=7$	0.823	0.787	0.881	0.845
$p=8$	0.850	0.822	0.890	0.849

TABLE 5.8
Comparison of time (sec.) $(p=6)$.

	TD(L)	PTD(L)	RTD(L)	PRTD(L)	TD(G)	PTD(G)	RTD(G)	PRTD(G)
CPU	362.5	301.3	258.4	215.1	472.1	411.7	373.8	321.5
Comm.	68.1	62.3	54.0	45.5	177.4	168.5	156.6	145.1
Comp.	278.1	250.2	198.5	174.3	279.5	252.1	200.1	176.7

TABLE 5.9
Comparison of time (sec.) $(p=8)$.

	TD(L)	PTD(L)	RTD(L)	PRTD(L)	TD(G)	PTD(G)	RTD(G)	PRTD(G)
CPU	402.5	345.3	300.4	272.4	530.6	500.6	445.1	403.2
Comm.	77.1	67.2	59.0	52.5	225.3	208.1	196.5	178.2
Comp.	305.2	265.4	214.2	192.0	308.5	268.0	215.1	193.5

to solve the systems of linear equation (5.9), we only use the TD, RTD, CG, PTD, PRTD, and the PCG methods to solve (5.9). We focus on BLOCK1 which is efficient for multiple superelements as seen in problem 2.

First we compare the number of iterations of the following methods: the CG, TD, RTD, PCG, PTD, and PRTD methods. The eight superelements are mapped onto 16 processors. The initial values are chosen randomly between 0 and 1; the stopping condition is $\|x^k - x\| \leq 10^{-6}$.

Table 5.6 shows the number of iterations in the RTD, PTD, TD, PTD, CG, and PCG methods for different values of p. From Table 5.6 we see that, for the L-shaped domain problem, TD methods converge faster than CG methods; PTD methods converge faster than PCG methods with the same preconditioner. Among the above methods we used, the PRTD method is the fastest convergence iterative method.

Table 5.7 shows the spectral radius of iteration matrices of the RTD, PRTD, TD, and PTD methods. From Table 5.7, the PRTD method has the smallest spectral radius among the above-mentioned four iterative methods. Our preconditioner accelerates the convergence of TD and RTD.

Tables 5.8 and 5.9 show the results of comparison of the execution time of TD, PTD, RTD, and PRTD for different values of p, using both local communication strategy and global communication strategy. It shows that PRTD has the best performance. Compared with the

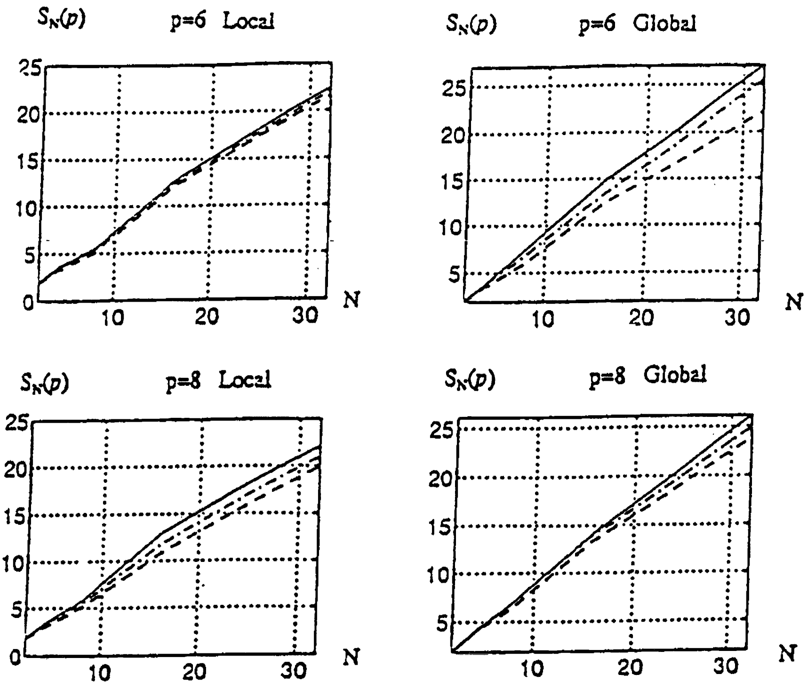


FIG. 5.7. *Speedup ratios.*

global communication strategy, local communication strategy saves more than 50% of communication time. L refers to local communication strategy; G refers to global communication strategy.

Now we apply the above iterative methods on 2^i, $i = 1, 2, 3, 4, 5$ processors and see the speedup ratio for each iterative method. Both global communication strategy and local communication strategy will be discussed. Figure 5.7 shows the speedups achievable for the RTD method (solid line), the TD method (dashed-dotted line), and the CG method (dashed line). Figure 5.8 shows the speedups achievable for the PRTD method, the PTD method (dashed-dotted line), and the PCG method (dashed line). $S_N(p)$ is the speedup ratio we defined in equation (4.16), and N is the number of processors.

From the numerical results of model problem 2 it is clear that the PRTD method with local communication strategy has the best performance in both computation and communication. The preconditioner accelerates the convergence for the CG, TD, and RTD methods. The PTD method converges faster than the PCG method with the same preconditioner. Local communication strategy saves almost two-thirds of the communication time required for the global communication.

6. Conclusions. Based on the numerical results from the two sample problems it is clear that the PRTD method using local communication strategy has the fastest convergence rate and best performance among the various iterative methods we discussed. The TD method converges faster than the CG method both with and without preconditioning. Local communication strategy saves not only the communication time but also the time required to assemble the global stiffness matrix and load vector. Superelement structure makes the TD method easy to map onto parallel computers (as shown here) and achieve high efficiency, i.e., computational load balance.

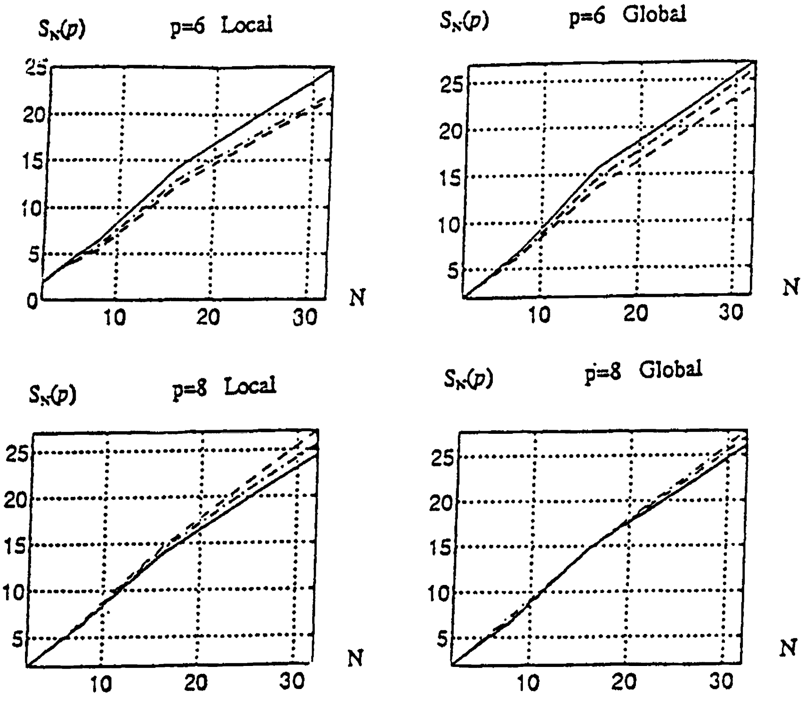


FIG. 5.8. *Speedup ratios (preconditioned).*

REFERENCES

[1] I. BABUSKA, B. A. SZABO, AND I. N. KATZ, *The p-version of the finite element method*, SIAM J. Numer. Anal., 18 (1981), pp. 515–545.

[2] B. A. SZABO AND I. BABUSKA, *Finite Element Analysis*, John Wiley & Sons, Inc. New York, 1991.

[3] I. BABUSKA AND H. C. ELMAN, *Some aspects of parallel implementation of the finite element method on message passing architecture*, J. Comput. Appl. Math., 27 (1989), pp. 157–187.

[4] I. BABUSKA, A. CRAIG, J. MANDEL, AND J. PITKARANTA, *Efficient Preconditioning for the p-version Finite Element Method in Two Dimensions*, Technical Report BN-1105, University of Maryland, College Park; Technical Report TE-41089, University of Colorado at Denver, 1989.

[5] G. BRUSSINO, R. HERBIN, Z. CHRISTIDIS, AND V. SONNAD, *Parallel Multilevel Finite Element Method with Hierachical Basis Functions*, IBM, Kingston, NY, manuscript.

[6] S. FORESTI, G. BRUSSINO, S. HASSANZADEH, AND V. SONNAD, *Multilevel Solution Method for the p-Version of Finite Elements*, Comput. Phys. Comm., 53 (1989), pp. 349–355.

[7] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computation*, The Johns Hopkins University Press, Baltimore, MD, 1989.

[8] W. D. GROPP AND D. E. KEYES, *A comparison of domain decomposition techniques for elliptic partial differential equations and their parallel implementation*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 166–202.

[9] ———, *Parallel Performance of Domain Decomposition Preconditioned Krylov Methods for PDE with Adaptive Refinement*, Technical rep. RR-773, Computer Science Department, Yale University, New Haven, CT, 1990.

[10] N. HU AND I. N. KATZ, *Multi-p Methods Based on Textured Decomposition Iterations*, presented at SIAM 40th Anniversary Meeting, Los Angeles, CA, July 1992.

[11] ———, *A Multi-p V-cycle Method Accelerated by a Nested Multi-p Procedure*, presented at SIAM Annual Meeting, Philadelphia, PA, July 1993.

[12] G. HUANG, W. K. TSAI, AND W. LU, *Fast parallel linear equation solver based on textured decomposition*, in IEEE Proc. 20th Conference on Decision and Control, 1987, pp. 1450–1454.

[13] J. MANDEL, *Adaptive Iterative Solvers in Finite Elements. Solving Large Scale Problems in Mechanics: Development and Application of Computational Solution Methods*, John Wiley and Sons, Ltd., New York, 1991.

[14] J. MANDEL, *Two-level domain decomposition preconditioning for the p-version finite element method in three dimensions*, Internat. J. Numer. Methods Engrg., 29 (1990), pp. 1095–1108.

[15] S. F. MCCORMICK, ED., *Multigrid Methods*, Vol. 3, Frontiers in Applied Mathematics, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1987.

[16] *NCUBE Handbook*, Version 1.0, NCUBE Corp., Beaverton, OR, April 1986.

[17] B. A. SZABO, *PROBE Theoretical Manual*, Release 1.0, Noetic Technologies Corp., St. Louis, MO, 1985.

[18] Y. ZHU AND I. N. KATZ, *A Parallel Implementation of the p-version of the Finite Element Method*, presented at Second ICIAM meeting, Washington DC, July 1991.

[19] ———, *Preconditioned Textured Decomposition for Parallel Implementation of the Finite Element Method*, presented at SIAM 40th Anniversary Meeting, Los Angeles, CA, July 1992.

[20] ———, *Recursive Textured Decomposition for the p-version of the Finite Element Method*, presented at SIAM Annual Meeting, Philadelphia, PA, July 1993.

SIAM J. SCI. COMPUT.
Vol. 17, No. 5, pp. 1068–1081, September 1996

MULTIGRID METHODS FOR SYMMETRIC POSITIVE DEFINITE BLOCK TOEPLITZ MATRICES WITH NONNEGATIVE GENERATING FUNCTIONS[*]

GIUSEPPE FIORENTINO[†] AND STEFANO SERRA[‡]

Abstract. In this paper we introduce a generalized multigrid method for solving linear systems $\mathbf{T}_{N,M}\mathbf{x} = \mathbf{b}$ where $\mathbf{T}_{N,M} \in \Re^{NM \times NM}$ is a symmetric block Toeplitz matrix with symmetric Toeplitz blocks. We use a special choice of the projection operator whose coefficients simply depend on the generating functions associated with the proposed class of matrices. This choice leads to iterative methods with convergence rates independent of the Euclidean condition number $\kappa_2(\mathbf{T}_{N,M})$ and of the dimension of the involved matrices. The total arithmetic cost is $O(NM\log(NM))$ for dense matrices and $O(NM)$ for band matrices with band blocks; in the PRAM model only $O(\log(NM))$ parallel steps are required. This algorithm is therefore competitive with the preconditioned conjugate gradient methods proposed by Chan and Jin [*SIAM J. Sci. Statist. Comput.*, 13 (1992), pp. 1218–1235], Ku and Kuo [*SIAM J. Sci. Statist. Comput.*, 13 (1992), pp. 948–966], Di Benedetto [*SIAM J. Sci. Comput.*, 17 (1996)], and Serra [*BIT*, 34 (1994), pp. 579–594] for dense matrices and improves those results for band block Toeplitz matrices with band Toeplitz blocks.

Key words. Toeplitz matrices, multigrid, iterative methods

AMS subject classification. 65F10

1. Introduction.

In this paper we introduce and discuss a generalized multigrid method [H1, H2] for the solution of linear systems associated with positive definite symmetric block Toeplitz matrices with symmetric Toeplitz blocks.

We consider the solution of the linear system of the form

$$\mathbf{T}_{N,M}\mathbf{x} = \mathbf{b}, \tag{1}$$

where

$$\mathbf{T}_{N,M} = \begin{bmatrix} \mathbf{T}_0 & \mathbf{T}_1 & \cdots & \mathbf{T}_{N-1} \\ \mathbf{T}_1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{T}_1 \\ \mathbf{T}_{N-1} & \cdots & \mathbf{T}_1 & \mathbf{T}_0 \end{bmatrix} \quad \text{and} \quad \mathbf{T}_i = \begin{bmatrix} a_{i,0} & a_{i,1} & \cdots & a_{i,M-1} \\ a_{i,1} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{i,1} \\ a_{i,M-1} & \cdots & a_{i,1} & a_{i,0} \end{bmatrix}, \tag{2}$$

with coefficients $a_{j,k}$ related to a continuous nonnegative function $a(x, y)$ as follows:

$$a_{j,k} = \frac{1}{\pi^2}\iint_Q a(x, y)e^{-i(jx+ky)}dxdy, \qquad Q = [0, \pi] \times [0, \pi].$$

The function $a(x, y)$ is called the generating function of $\mathbf{T}_{N,M}$. All the eigenvalues of $\mathbf{T}_{N,M}$ belong to the range of $a(x, y)$ and, moreover (see [S2]), we have

$$\lim_{N,M\to\infty} \lambda_{\max}(\mathbf{T}_{N,M}) = \max_Q a(x, y) \quad \text{and} \quad \lim_{N,M\to\infty} \lambda_{\min}(\mathbf{T}_{N,M}) = \min_Q a(x, y).$$

Consequently, when the function $a(x, y)$ is nonnegative and vanishes at some point of Q, we find that $\mathbf{T}_{N,M}$ is ill conditioned for any value of N and M. More precisely, the Euclidean condition number of $\mathbf{T}_{N,M}$, as a function of the dimensions, is unbounded:

$$\lim_{N,M\to\infty} \kappa_2(\mathbf{T}_{N,M}) = +\infty.$$

[*] Received by the editors July 20, 1994; accepted for publication (in revised form) March 2, 1995.

[†] Dipartimento di Informatica, Università of Pisa, Corso Italia 40, 56100 Pisa, Italy (fiorent@di.unipi.it).

[‡] Dipartimento di Matematica, Università of Milano, Via C. Saldini 50, 20133 Milano, Italy.

Thus, unless some preconditioning is used, all classic iterative methods are very slow.

The state of the art in the most recent literature is the following.

(a) *Direct methods*:

(1) *Fast methods* with arithmetic cost of $O(N^2M^3)$ [Tr],

(2) *Superfast methods* with arithmetic cost of $O(NM^3\log^2(N))$ [Mu, Ri].

These methods are not suitable for implementation in a parallel model of computation since they are intrinsically sequential; this problem is not encountered using iterative methods.

(b) *Preconditioned conjugate gradient methods*:

(1) Using a preconditioner in the two-level circulant algebra, [Da, KK], and [CJ] obtain an iterative method with an overall arithmetic cost of $O(NM(N+M)\log(NM))$, but in actual computations the cost is generally $O(NM\log(NM))$, i.e., the cost of a bivariate fast Fourier transform [BP].

(2) The same result is reported in [DB] where preconditioners have been chosen in the two-level τ algebra [BC].

(3) A cost of $O(NM\log(NM))$ is proved in [S2] by means of block band Toeplitz preconditioners with band blocks.

In these cases, a parallel cost of $O(\log(NM))$ is found in practice; for b.3 it is also proved.

In this paper, at first we devise a multigrid algorithm for block-τ systems; then we extended it to the Toeplitz case. The main features are the following:

- The overall arithmetic cost is $O(NM\log(NM))$ in the sequential model and $O(\log(NM))$ in the parallel case. Moreover, when $\mathbf{T}_{N,M}$ is a band block Toeplitz matrix with band blocks, our algorithm achieves a linear cost of $O(NM)$ arithmetic operations, while methods b.1, b.2, and b.3 lead to the same arithmetic cost of $O(NM\log(NM))$. This feature makes our method also profitable as an efficient preconditioner solver for methods like b.3.
- The choice of a projector, the most delicate step in a multigrid scheme, is straightforward in our method since, in this case, the projector simply depends on some properties of the generating function $a(x,y)$.

The main idea is to define a projector operator in terms of the bivariate function associated with the matrix; in fact, we only require the knowledge of the zeros of $a(x,y)$ and their multiplicity to devise a rapidly convergent method.

The paper is organized as follows: in §2 we introduce the two-grid method; in §3 we describe the τ algebra [BC], its connections with Toeplitz matrices, and a special kind of two-grid method (TGM) for these matrices.

In §4 we perform a structural analysis of the two-grid iteration matrix that, in §5, allows us to devise a class of projector operators which guarantee a convergence rate independent of the size of the related linear system.

In §6 we exploit the close spectral similarity of τ and Toeplitz matrices applying our multigrid technique to Toeplitz systems. All numerical results reported in §6 confirm the effectiveness of the method.

2. Overview of a TGM. Given an $n\times n$ linear system $\mathbf{Ax}=\mathbf{b}$, an iterative method for its solution

$$\mathbf{z}^{(i+1)}=\mathbf{Sz}^{(i)}+\mathbf{c}=\mathcal{S}_{\mathbf{c}}(\mathbf{z}^{(i)}), \tag{3}$$

and a $k\times n$ $(k<n)$ matrix $\mathbf{p}$, a general TGM is defined by the following steps [H1, Gr]:

Step 1. $\mathbf{x}^{(i,\sigma)}=\mathcal{S}_{\mathbf{c}}^{\sigma}(\mathbf{x}^{(i)})$,

Step 2. $\mathbf{d}_n=\mathbf{Ax}^{(i,\nu)}-\mathbf{b}$,

Step 3. $\mathbf{d}_k=\mathbf{pd}_n$,

Step 4. $\mathbf{A}_k = \mathbf{pAp}^T$,
Step 5. Solve $\mathbf{A}_k\mathbf{y} = \mathbf{d}_k$,
Step 6. $\mathbf{x}^{(i+1)} = \mathbf{x}^{(i,\sigma)} - \mathbf{p}^T\mathbf{y}$.

Step 1, the "smoothing iteration" in the literature, consists of σ sweeps of the iterative method (3). For a broad class of methods like (3), after a few iterations the error $\mathbf{d}_n = \mathbf{Ax}^{(i,\sigma)} - \mathbf{b}$ is typically confined in a subspace where the method converges slowly. When this subspace is known, it is possible to accelerate the convergence by means of a "coarse grid correction" that computes and subtracts from $\mathbf{x}^{(i,\sigma)}$ an approximation of the error $\mathbf{d}_n$.

Steps 2–6 define this coarse grid correction: the error is first multiplied by the rectangular $k \times n$ matrix $\mathbf{p}$ to obtain the projected error $\mathbf{d}_k = \mathbf{pd}_n$; then a "correction" is calculated on a k-dimensional subspace strongly related to the subspace where (3) is slowly contractive.

The global iteration matrix of a TGM is therefore given by

$$TGM(\mathbf{S}, \mathbf{p}) = \left[\mathbf{I}_n - \mathbf{p}^T(\mathbf{pAp}^T)^{-1}\mathbf{pA}\right]\mathbf{S}^\sigma, \tag{4}$$

where the expression within brackets is related to the coarse grid correction while the rightmost matrix comes from the σ sweeps of (3).

In order to define a fast generalized TGM for block Toeplitz problems, we need to find a matrix $\mathbf{p}$ that better exploits the spectral features of $\mathbf{A}$ and $\mathbf{S}$. The objective is to arrive at a spectral radius of the matrix TGM(**S**,**p**) smaller than 1 and independent of the dimension n.

3. The TGM for τ matrices. In this section we introduce a generalized TGM for block τ matrices, but first we recall some important facts about τ matrices and their strong relations with the Toeplitz class.

The two-level algebra $\tau_{N,M}$ is generated by the matrices $\mathbf{H}_n \otimes \mathbf{I}_M$ and $\mathbf{I}_N \otimes \mathbf{H}_M$, where

$$\mathbf{H}_k = \begin{bmatrix} 0 & 1 & & O \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ O & & 1 & 0 \end{bmatrix} \in \Re^{k\times k}$$

and $\otimes$ denotes the Kronecker product defined by $[a_{ij}]_{ij} \otimes \mathbf{B} = [a_{ij}\mathbf{B}]_{ij}$.

Since $\mathbf{H}_k$ is diagonalized by

$$\mathbf{H}_k = 2\mathbf{F}_k \begin{bmatrix} \cos(x_1^{(k)}) & & O \\ & \ddots & \\ O & & \cos(x_k^{(k)}) \end{bmatrix} \mathbf{F}_k, \quad \text{where } x_i^{(k)} = \frac{i\pi}{k+1},$$

and $\mathbf{F}_k = [\mathbf{f}_1 | \dots | \mathbf{f}_k]$ with $\mathbf{f}_i = \sqrt{\frac{2}{k+1}}\,[\sin\frac{ij\pi}{k+1}]_{j=1}^k$, it follows that the eigenvectors of any $\tau_{N,M}$ matrix are given by the columns of $\mathbf{F}_N \otimes \mathbf{F}_M$.

There is a strong link between $\tau_{N,M}$ and the Toeplitz class; namely, given the Toeplitz matrix

$$\mathbf{T} = \begin{bmatrix} a_0 & a_1 & \cdots & a_{k-1} \\ a_1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_1 \\ a_{k-1} & \cdots & a_1 & a_0 \end{bmatrix}$$

and the Hankel operator

$$H(\mathbf{T}) = \begin{bmatrix} a_2 & \cdots & a_{k-1} & 0 & 0 \\ \vdots & \cdot^{\cdot^{\cdot}} & \cdot^{\cdot^{\cdot}} & \cdot^{\cdot^{\cdot}} & 0 \\ a_{k-1} & \cdot^{\cdot^{\cdot}} & \cdot^{\cdot^{\cdot}} & \cdot^{\cdot^{\cdot}} & a_{k-1} \\ 0 & \cdot^{\cdot^{\cdot}} & \cdot^{\cdot^{\cdot}} & \cdot^{\cdot^{\cdot}} & \vdots \\ 0 & 0 & a_{k-1} & \cdots & a_2 \end{bmatrix},$$

a $\tau_{N,M}$ approximation $\mathbf{A}_{N,M}$ of $\mathbf{T}$ is obtained by the following two steps [DB]:

(1) for each block $\mathbf{T}_i$ in (2), define $\mathbf{A}_i = \mathbf{T}_i - H(\mathbf{T}_i)$ and

$$\tilde{\mathbf{T}} = \begin{bmatrix} \mathbf{A}_0 & \mathbf{A}_1 & \cdots & \mathbf{A}_{N-1} \\ \mathbf{A}_1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{A}_1 \\ \mathbf{A}_{N-1} & \cdots & \mathbf{A}_1 & \mathbf{A}_0 \end{bmatrix};$$

(2) define $\mathbf{A}_{N,M} = \tilde{\mathbf{T}} - H(\tilde{\mathbf{T}})$ in terms of blocks.

The matrix $\mathbf{A}_{N,M} \in \tau_{N,M}$ is closely related to $\mathbf{T}_{N,M}$; in fact its eigenvalues $\alpha_{\mu,\nu}$ are obtained by sampling an approximation of the generating function of $\mathbf{T}_{N,M}$:

$$(5) \qquad \alpha_{\mu,\nu} = \mathrm{Trunc}(a, N-1, M-1)\big|_{(x_\mu^{(N)}, x_\nu^{(M)})},$$

where $\mathrm{Trunc}(a, p, q)$ is the bivariate trigonometric polynomial of degree p and q given by the first terms of the Fourier expansion of $a(x, y)$.

For the solution of the linear system $\mathbf{A}_{N,M}\mathbf{x} = \mathbf{b}$ we use the iterative method of the form

$$(6) \qquad \mathbf{x}^{(i+1)} = (\mathbf{I} - \mathbf{A}_{N,M}/\alpha)\mathbf{x}^{(i)} + \mathbf{b}/\alpha, \quad \text{where } \alpha = \max_{Q} a(x, y).$$

Since the Fourier transform is a linear operator and the iteration matrix $\mathbf{S}_{N,M} = \mathbf{I} - \mathbf{A}_{N,M}/\alpha$ of (6) is an affine transformation of $\mathbf{A}_{N,M}$, it follows that the generating function of $\mathbf{S}_{N,M}$ is given by $s(x, y) = 1 - a(x, y)/\alpha$. The value of $s(x, y)$ is therefore close to one when $a(x, y)$ is close to zero; consequently (6) is slowly convergent on the subspaces generated by the eigenvectors of $\mathbf{A}_{N,M}$ corresponding to eigenvalues close to zero.

We obtain an efficient algorithm introducing a TGM for which a suitable coarse grid correction takes care of the slowly convergent subspaces. The search for a fitting projector $\mathbf{p}$ will be solved in functional terms.

First we show how to obtain a projector from a $\tau_{N,M}$ matrix $\mathbf{P}_{N,M}$ generated by a generic function $p(x, y) \geq 0$. In the next sections we explain how to get a profitable function $p(x, y)$ to achieve a convergence rate independent of the dimension of the matrices.

Given $\mathbf{P}_{N,M}$, assuming $N = 2n + 1$ and $M = 2m + 1$, we set

$$(7) \qquad \mathbf{p} = \mathbf{K}_{N,M}\mathbf{P}_{N,M},$$

where $\mathbf{K}_{N,M}$ is the cutting matrix [FS] in two dimensions defined by

$$\mathbf{K}_{N,M} = \mathbf{K}_N \otimes \mathbf{K}_M = \begin{bmatrix} 0 & \mathbf{K}_M & 0 & \cdots & & & O \\ 0 & 0 & 0 & \mathbf{K}_M & 0 & \cdots & \\ & & & & \ddots & \ddots & \ddots \\ O & & & \cdots & 0 & \mathbf{K}_M & 0 \end{bmatrix} = [\Re^{m\times M}]^{n\times N}$$

with

$$\mathbf{K}_M = \begin{bmatrix} 0 & 1 & 0 & \cdots & & & & O \\ 0 & 0 & 0 & 1 & 0 & \cdots & & \\ & & & & \ddots & \ddots & \ddots & \\ O & & & & \cdots & 0 & 1 & 0 \end{bmatrix} \in \Re^{m\times M}.$$

In this way we easily obtain a two-grid iteration matrix like (4):

$$TGM(\mathbf{S},\mathbf{p}) \equiv [\mathbf{I} - \mathbf{p}^T(\mathbf{A}_{n,m})^{-1}\mathbf{p}\mathbf{A}_{N,M}]\mathbf{S}^{\sigma}_{N,M}, \tag{8}$$

where $\mathbf{A}_{n,m} \equiv \mathbf{p}\mathbf{A}_{N,M}\mathbf{p}^T$ is the reduced problem matrix related to the course grid correction operator

$$CGC(\mathbf{p}) \equiv [\mathbf{I} - \mathbf{p}^T(\mathbf{A}_{n,m})^{-1}\mathbf{p}\mathbf{A}_{N,M}].$$

4. Spectral properties. Now we introduce some tools that will be useful for analyzing the convergence of our TGM. A full spectral analysis of the matrices involved in the algorithm will be given along with a complete block decomposition. For the sake of simplicity, and without loss of generality, we assume $\sigma = 1$.

Rearranging the columns of $\mathbf{F}_N$ we define

$$\hat{\mathbf{F}}_N = [\mathbf{f}_1|\mathbf{f}_N|\ldots|\mathbf{f}_i|\mathbf{f}_{N-i+1}|\ldots|\mathbf{f}_n]$$

and $\hat{\mathbf{F}}_M$ analogously. The matrix

$$\mathbf{V}_N = \left[\begin{array}{cc|cc|cc|cc|c} 1 & -1 & & & & & & & \\ \hline & & 1 & -1 & & & & & \\ \hline & & & & \ddots & \ddots & & & \\ \hline & & & & & & 1 & -1 & \\ \hline & & & & & & & & 0 \end{array}\right]$$

allows us to relate the cutting matrix and the projector as follows:

$$\mathbf{K}_N\hat{\mathbf{F}}_N = \sqrt{2}\,\mathbf{F}_n\mathbf{V}_N. \tag{9}$$

The reduced problem matrix $\mathbf{A}_{n,m} = \mathbf{p}\mathbf{A}_{N,M}\mathbf{p}^T$, in view of (7) and (9), can be therefore written as

$$\begin{aligned} \mathbf{p}\mathbf{A}_{N,M}\mathbf{p}^T &= (\mathbf{K}_N \otimes \mathbf{K}_M)\mathbf{P}_{N,M}\mathbf{A}_{N,M}\mathbf{P}_{N,M}(\mathbf{K}_N \otimes \mathbf{K}_M)^T \\ &= (\mathbf{K}_N \otimes \mathbf{K}_M)(\hat{\mathbf{F}}_N \otimes \hat{\mathbf{F}}_M)\hat{\Lambda}_{\mathbf{p}}\hat{\Lambda}_{\mathbf{A}}\hat{\Lambda}_{\mathbf{p}}(\hat{\mathbf{F}}_N \otimes \hat{\mathbf{F}}_M)(\mathbf{K}_N \otimes \mathbf{K}_M)^T \\ &= 4(\mathbf{F}_n \otimes \mathbf{F}_m)(\mathbf{V}_N \otimes \mathbf{V}_M)\hat{\Lambda}_{\mathbf{p}}\hat{\Lambda}_{\mathbf{A}}\hat{\Lambda}_{\mathbf{p}}(\mathbf{V}_N \otimes \mathbf{V}_M)^T(\mathbf{F}_n \otimes \mathbf{F}_m) \\ &\equiv 4(\mathbf{F}_n \otimes \mathbf{F}_m)\hat{\mathbf{A}}_{n,m}(\mathbf{F}_n \otimes \mathbf{F}_m), \end{aligned} \tag{10}$$

where

$$\hat{\Lambda}_{\mathbf{A}} = \left[\begin{array}{c|c} \operatorname*{diag}_{1\le\mu\le n}\begin{bmatrix} \hat{\Lambda}_{\mathbf{A},\mu} & O \\ O & \hat{\Lambda}_{\mathbf{A},\mu'} \end{bmatrix} & 0 \\ \hline O & \hat{\Lambda}_{A,n} \end{array}\right],$$

$$\hat{\Lambda}_{\mathbf{A},\mu} = \left[\begin{array}{c|c} \operatorname*{diag}_{1\le\nu\le m}\begin{bmatrix} \alpha_{\mu,m} & 0 \\ 0 & \alpha_{\mu,\nu'} \end{bmatrix} & O \\ \hline O & \alpha_{\mu,m} \end{array}\right]$$

with $\mu' = N - \mu + 1$ and $\nu' = M - \nu + 1$. $\hat{\Lambda}_{\mathbf{p}}$ is defined in the same way.

A generic block of $\hat{\mathbf{A}}_{n,m} = (\mathbf{V}_N \otimes \mathbf{V}_M)\hat{\Lambda}_{\mathbf{p}}\hat{\Lambda}_{\mathbf{A}}\hat{\Lambda}_{\mathbf{p}}(\mathbf{V}_N \otimes \mathbf{V}_M)^T$ in (10) is consequently given by

$$[\mathbf{V}_M \quad -\mathbf{V}_M]\begin{bmatrix} \hat{\Lambda}_{\mathbf{A},\mu}\hat{\Lambda}^2_{\mathbf{p},\mu} & 0 \\ 0 & \hat{\Lambda}_{\mathbf{A},\mu'}\hat{\Lambda}^2_{\mathbf{p},\mu'} \end{bmatrix}\begin{bmatrix} \mathbf{V}^T_M \\ -\mathbf{V}^T_M \end{bmatrix}.$$

It follows that $\hat{\mathbf{A}}_{n,m} = \operatorname{diag}_{1\le\mu\le n}[\operatorname{diag}_{1\le\nu\le m}[\hat{\alpha}_{\mu,\nu}]]$, where

$$\hat{\alpha}_{\mu,\nu} = \alpha_{\mu,\nu}p^2_{\mu,\nu} + \alpha_{\mu,\nu'}p^2_{\mu,\nu'} + \alpha_{\mu'\nu}p^2_{\mu',\nu} + \alpha_{\mu',\nu'}p^2_{\mu',\nu'}$$

and $p_{\mu,\nu} = \operatorname{Trunc}(p, N-1, M-1)|_{(x^{(N)}_\mu, x^{(M)}_\nu)}$ (compare (5)).

The spectral block decomposition of the coarse grid correction operator is obtained in a similar way:

$$CGC(\mathbf{p}) = \mathbf{F}_N \otimes \mathbf{F}_M \cdot \operatorname*{diag}_{1\le\mu\le n}\left[\operatorname*{diag}_{1\le\nu\le m}[C_{\mu,\nu}]\right] \cdot \mathbf{F}_n \otimes \mathbf{F}_M,$$

where

$$C_{\mu,\nu} = \mathbf{I}_4 - \underbrace{\begin{bmatrix} p_{\mu,\nu} \\ -p_{\mu,\nu'} \\ p_{\mu'\nu} \\ -p_{\mu',\nu'} \end{bmatrix}\frac{1}{\hat{\alpha}_{\mu,\nu}}[p_{\mu,\nu} \quad -p_{\mu,\nu'} \quad p_{\mu',\nu} \quad -p_{\mu',\nu'}]}_{(*)}$$

$$\times \underbrace{\begin{bmatrix} \alpha_{\mu,\nu} & & & O \\ & \alpha_{\mu,\nu'} & & \\ & & \alpha_{\mu',\nu} & \\ O & & & \alpha_{\mu',\nu'} \end{bmatrix}}_{(**)}.$$

Thus, the matrix $\hat{\mathbf{F}}_N \otimes \hat{\mathbf{F}}_M \cdot TGM(\mathbf{p}) \cdot \hat{\mathbf{F}}_N \otimes \hat{\mathbf{F}}_M$ is diagonalized into nm 4×4 blocks and $N+M-1$ scalar zeros; the spectral radius of $TGM(\mathbf{S}, \mathbf{p})$ is therefore given by $\sup_{\mu,\nu}\{\rho(M_{\mu,\nu})\}$ where

$$M_{\mu,\nu} = C_{\mu,\nu}\begin{bmatrix} s_{\mu,\nu} & & & O \\ & s_{\mu,\nu'} & & \\ & & s_{\mu',\nu} & \\ O & & & s_{\mu',\nu'} \end{bmatrix}$$

and $s_{\mu,\nu} = \operatorname{Trunc}(N-1, M-1, s)|_{(x^{(N)}_\mu, x^{(M)}_\nu)}$ (compare (5)).

From [Gr], we know that (*) is the inverse of (**) in the linear subspace generated by $\hat{p} = [p_{\mu,\nu} \quad -p_{\mu,\nu'} \quad p_{\mu',\nu} \quad -p_{\mu',\nu'}]^T$ and represents the null matrix in the three-dimensional orthogonal subspace. The product of (*) and (**) is therefore the identity operator in $\mathbf{span}(\hat{p})$ and the null operator in $\mathbf{span}(\hat{p})^\perp$.

It follows that the eigenvalues of $C_{\mu,\nu}$ are zero with multiplicity one and one with multiplicity three; as a result, one eigenvalue of $M_{\mu,\nu}$, say $\lambda_1^{\mu,\nu}$, is equal to zero, but further considerations are needed to evaluate the other three: $\lambda_2^{\mu,\nu}$, $\lambda_3^{\mu,\nu}$, and $\lambda_4^{\mu,\nu}$.

Recalling that both $\mathbf{A}$ and $\mathbf{S} \equiv \mathbf{I} - \mathbf{A}/\alpha$ are positive definite and that they commute (because $\mathbf{S}$ is a polynomial of $\mathbf{A}$), we find that $TGM(\mathbf{S}, \mathbf{p})$ is similar to

$$\begin{aligned} TGM^*(\mathbf{S}, \mathbf{p}) &= \mathbf{S}^{1/2}\mathbf{A}^{1/2} TGM(\mathbf{p})\mathbf{A}^{-1/2}\mathbf{S}^{-1/2} \\ &= \mathbf{S} - \mathbf{S}^{1/2}\mathbf{A}^{1/2}\mathbf{p}^T(\mathbf{p}\mathbf{A}\mathbf{p}^T)^{-1}\mathbf{p}\mathbf{A}\mathbf{S}\mathbf{A}^{-1/2}\mathbf{S}^{-1/2} \\ &= \mathbf{S} - \mathbf{S}^{1/2}\mathbf{A}^{1/2}\mathbf{p}^T(\mathbf{p}\mathbf{A}\mathbf{p}^T)^{-1}\mathbf{p}\mathbf{A}^{1/2}\mathbf{S}^{1/2}, \end{aligned}$$

which is clearly symmetric; hence, $\lambda_i^{\mu,\nu} \in \Re$ for $i = 1, \dots, 4$.

From the previous equation it also follows that

$$TGM^*(\mathbf{p}) = \mathbf{S}^{1/2}(\mathbf{I} - \mathbf{A}^{1/2}\mathbf{p}^T(\mathbf{p}\mathbf{A}\mathbf{p}^T)^{-1}\mathbf{p}\mathbf{A}^{1/2})\mathbf{S}^{1/2}$$

and, since $\mathbf{I} - \mathbf{A}^{1/2}\mathbf{p}^T(\mathbf{p}\mathbf{A}\mathbf{p}^T)^{-1}\mathbf{p}\mathbf{A}^{1/2}$ is an orthogonal projector operator, its eigenvalues belong to the set $\{0, 1\}$; see [Gr]. Therefore, for any $\mathbf{x} \in \Re^{NM}$ setting $\mathbf{y} = \mathbf{S}^{1/2}\mathbf{x}$, by means of the Rayleigh quotient, we obtain

$$(11) \qquad 0 \le \frac{\mathbf{x}^T TGM^*(\mathbf{p})\mathbf{x}}{\mathbf{x}^T\mathbf{x}} = \frac{\mathbf{y}^T(\mathbf{I} - \mathbf{A}^{1/2}\mathbf{p}^T(\mathbf{p}\mathbf{A}\mathbf{p}^T)^{-1}\mathbf{p}\mathbf{A}^{1/2})\mathbf{y}}{\mathbf{x}^T\mathbf{x}} \le \frac{\mathbf{y}^T\mathbf{y}}{\mathbf{x}^T\mathbf{x}} = \frac{\mathbf{x}^T\mathbf{S}\mathbf{x}}{\mathbf{x}^T\mathbf{x}}.$$

Finally, from (11), we have

$$0 \le \lambda_i^{\mu,\nu} \le \rho(\mathbf{S}) < 1 \quad \text{for } i = 1, \dots, 4.$$

5. The choice of p. Inequality (11) shows that, for any choice of $\mathbf{p}$, the multigrid method outlined in the previous section is never slower than the relaxation method (6) alone.

We recall that the spectral decomposition (10) holds for τ matrices; for Toeplitz matrices it can be proved that (10) is true only in an approximate sense. On the other hand, inequality (11) is completely general.

In the following, without loss of generality, we assume that $0 \le a(x, y) \le 1$ and the existence of a point $z_0 = (x_0, y_0) \in [0, \pi]^2$ such that $a(x, y)$ vanishes in z_0 and is positive elsewhere. This zero leads to numerical difficulties in the resolution of (1) for any N and M since, for any N and M, we may find indexes μ_N and ν_n such that

$$\lim_{N,M \to +\infty} (x_{\mu_N}^{(N)}, x_{\nu_M}^{(M)}) = z_0, \quad \text{where } x_i^{(k)} = \frac{i\pi}{k+1}.$$

To analyze the convergence of $TGM(\mathbf{S}, \mathbf{p})$ as given in (8), we assume for brevity $\mu = \mu_n$, $\nu = \nu_m$ and analyzé the behavior in the worst subspace.

From

$$\mathbf{y}_{N,M} = M_{\mu,\nu} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

if we set $\mathbf{S} = \mathbf{I} - \mathbf{A}$ in (8), we find that

$$\mathbf{y}_{N,M} = CGC(\mathbf{p})(1-\alpha_{\mu,\nu})\begin{bmatrix}1\\0\\0\\0\end{bmatrix} = (1-\alpha_{\mu,\nu})\left[\begin{bmatrix}1\\0\\0\\0\end{bmatrix} - \frac{\hat{\mathbf{p}}\cdot\hat{\mathbf{p}}^T}{\hat{\alpha}_{\mu,\nu}}\begin{bmatrix}\alpha_{\mu,\nu}\\0\\0\\0\end{bmatrix}\right]$$

$$= \frac{1-\alpha_{\mu,\nu}}{\hat{\alpha}_{\mu,\nu}}\begin{bmatrix}\alpha_{\mu,\nu'}p^2_{\mu,\nu'} + \alpha_{\mu',\nu}p^2_{\mu',\nu} + \alpha_{\mu',\nu'}p^2_{\mu',\nu'}\\ \alpha_{\mu,\nu}p_{\mu,\nu}p_{\mu,\nu'}\\ -\alpha_{\mu,\nu}p_{\mu,\nu}p_{\mu',\nu}\\ \alpha_{\mu,\nu}p_{\mu,\nu}p_{\mu',\nu'}\end{bmatrix}.$$

Recalling that $\hat{a}_{\mu,\nu} = a_{\mu,\nu}p^2_{\mu,\nu} + a_{\mu,\nu'}p^2_{\mu,\nu'} + a_{\mu',\nu'}p^2_{\mu',\nu'} + a_{\mu',\nu'}p^2_{\mu',\nu'}$, if we choose an arbitrary function $p(x, y) > 0$, from $\lim_{N,M\to+\infty}\alpha_{\mu,\nu} = a(z_0) = 0$, we obtain

$$\lim_{N,M\to+\infty} M_{\mu,\nu}\begin{bmatrix}1\\0\\0\\0\end{bmatrix}^T = \begin{bmatrix}1\\0\\0\\0\end{bmatrix},$$

that is, a $TGM(\mathbf{S}, \mathbf{p})$ with a spectral radius close to one.

To improve the contractivity we need a function $p(x, y)$ such that

$$\limsup_{N,M\to+\infty}\left\{\frac{p^2_{\mu,\nu'}}{\alpha_{\mu,\nu}}, \frac{p^2_{\mu',\nu}}{\alpha_{\mu,\nu}}, \frac{p^2_{\mu',\nu'}}{\alpha_{\mu,\nu}}\right\} < +\infty;$$

this implies that

$$\limsup_{N,M\to+\infty}\frac{\alpha_{\mu,\nu'}p^2_{\mu,\nu'} + \alpha_{\mu',\nu}p^2_{\mu',\nu} + \alpha_{\mu',\nu'}p^2_{\mu',\nu'}}{\hat{\alpha}_{\mu,\nu}} = \rho < 1. \tag{12}$$

In this case we have $\|\mathbf{y}_{N,M}\|_2^2 = (1-\alpha_{\mu,\nu})^2(\chi_{N,M} + \gamma_{N,M})$, where

$$\chi_{N,M} = \left(\frac{\alpha_{\mu,\nu'}p^2_{\mu,\nu'} + \alpha_{\mu',\nu}p^2_{\mu',\nu} + \alpha_{\mu',\nu'}p^2_{\mu',\nu'}}{\hat{\alpha}_{\mu,\nu}}\right)^2 \quad \text{and}$$

$$\gamma_{N,M} = \frac{\alpha^2_{\mu,\nu}p^2_{\mu,\nu}(p^2_{\mu,\nu'} + p^2_{\mu',\nu} + p^2_{\mu',\nu'})}{\hat{\alpha}^2_{\mu,\nu}}.$$

Then, for $N, M \to +\infty$, we find $\chi_{N,M} \to \rho^2 < 1$, $(1-\alpha_{\mu,\nu})^2 \to 1$, and $\gamma_{N,M} \to 0$. Consequently we obtain $\|y_{N,M}\|_2^2 \to \rho^2 < 1$ and a $TGM(\mathbf{S}, \mathbf{p})$ strongly contractive in the subspace where $\mathbf{S}$ is a slowly convergent iteration matrix.

Condition (12) gives enough information to construct a projector operator that guarantees a convergence speed independent of the dimensions N and M; it states that whenever $a(x, y)$ has a minimum in (x_0, y_0), we have to choose a projector among those vanishing at the "mirror" points $(\pi - x_0, y_0)$, $(x_0, \pi - y_0)$, and $(\pi - x_0, \pi - y_0)$. See Fig. 1.

The trigonometric polynomial

$$\begin{aligned}p(x, y) &= [\cos(2x) - \cos(2x_0) + \cos(y) - \cos(\pi - y_0)]\\ &\quad\times [\cos(x) - \cos(\pi - x_0) + \cos(2y) - \cos(2y_0)]\end{aligned}$$

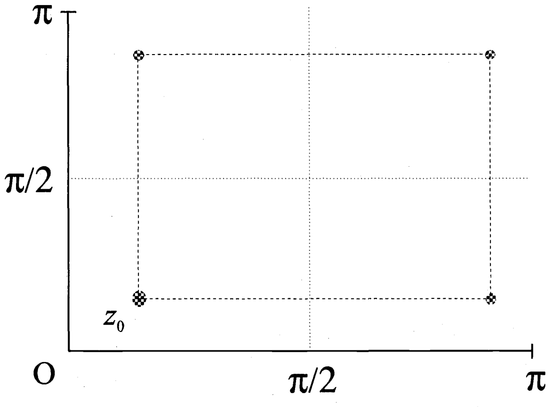


FIG. 1. *"Mirror" points for* z_0.

is a convenient choice for the function $p(x, y)$ because, ignoring any constant factor, we may easily find an associated matrix $\mathbf{P}$ in the Toeplitz class; in fact, setting

$$\mathbf{C}_1^{(\theta)} = \begin{bmatrix} -2\cos(\theta) & 1 & & & O \\ 1 & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ O & & & 1 & -2\cos(\theta) \end{bmatrix} \quad \text{and}$$

$$\mathbf{C}_2^{(\theta)} = \begin{bmatrix} -2\cos(2\theta) & 0 & 1 & & & O \\ 0 & \ddots & \ddots & \ddots & & \\ 1 & \ddots & \ddots & \ddots & & 1 \\ & \ddots & \ddots & \ddots & & 0 \\ O & & & 1 & 0 & -2\cos(2\theta) \end{bmatrix},$$

the matrix $\mathbf{P}$ can be written as

$$\mathbf{P} = \mathbf{P}(x_0, y_0) \equiv (\mathbf{C}_2^{(x_0)} \otimes \mathbf{I}_M + \mathbf{I}_N \otimes \mathbf{C}_1^{(\pi - y_0)}) \cdot (\mathbf{C}_1^{(\pi - x_0)} \otimes \mathbf{I}_M + \mathbf{I}_N \otimes \mathbf{C}_2^{(y_0)}).$$

Moreover, when we find $a(z_0) = 0$ with multiplicity $d > 1$, i.e.,

$$0 < \limsup_{z \to z_0} \frac{a(z)}{\|z - z_0\|_2^d} < +\infty,$$

we have to choose $p(x, y)$ such that its zeros at the mirror points have multiplicity at least $d/2$ to obtain a rapidly convergent method. In this case, we may use $\mathbf{P} \equiv \mathbf{P}(z_0)^{\lceil d/2 \rceil}$.

6. Numerical results. We now present some numerical results obtained by applying our generalized multigrid method to ill-conditioned block symmetric Toeplitz matrices of increasing difficulty. All the generating functions associated with these problems are equal to zero at some point z_0. In all cases the knowledge of z_0 and its multiplicity has been enough to devise a convergent algorithm.

We have to recall that the projector operators are constructed by considering $\mathbf{A}_{N,M}$; therefore, much of the convergence analysis performed in the previous sections only concerns τ matrices. In this section, by using the strong spectral similarity between τ and Toeplitz matrices, we apply our algorithm to Toeplitz systems like (1). Also in this case we obtain fast multigrid methods as shown by the numerical tests in the following sections.

6.1. The framework. In the following, we consider linear systems $\mathbf{Tx} = \mathbf{b}$, where $\mathbf{T}$ is a square 400×400 matrix, i.e.; $N = M = 20$. (We have performed numerical experiments with different values of N and M, but we report only this case since, when the dimension changes, the number of required iterations remains almost the same.) For all tests, starting from a randomly chosen vector, we report the 2-norm of the error at each iteration. All numerical computations have been carried out with double precision arithmetic.

We choose the projectors according to the theoretic considerations of the preceding sections obtaining spectral radii bounded by constants less than one and independent of the dimension of the matrices. As validated by the numerical experiments in this section, at each step the error is reduced by a constant factor that is independent of N and M.

We will use smoothers taken from amongst the relaxed methods of the form

$$\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - (\mathbf{Ax} - \mathbf{b})/\alpha.$$

Our numerical experiments have been performed using two smoothers (called pre- and postsmoother) as proposed in [FS]. More precisely, setting $\alpha = \|a\|_\infty$ that is asymptotically close to $\|\mathbf{A}\|_2$, we define the presmoother iteration matrix as

$$\mathbf{S}_1 = \mathbf{I} - \mathbf{A}/\alpha, \tag{13}$$

while the postsmoother is

$$\mathbf{S}_2 = \mathbf{I} - 2\mathbf{A}/\alpha, \tag{14}$$

The method is therefore a multi-iterative one [S1], where the presmoother approaches the exact solution in a fixed subspace then the coarse grid correction approaches it in the complementary one. The postsmoother $\mathbf{S}_2$ accelerates the overall convergence rate reducing the error in a third intermediate subspace where neither $\mathbf{S}_1$ nor the coarse grid correction is highly contractive. In Tables 1–4 the parameters `Pre` and `Post` denote the number of the pre- and postsmoothing sweeps made at each global iteration.

6.2. The Poisson problem on the unit square. The first example is a classical one, the Poisson problem on the unit square $\Omega = [0, 1] \times [0, 1]$:

$$\begin{cases} \Delta f(x, y) = g(x, y), & (x, y) \in \Omega, \\ f(x, y) = 0, & (x, y) \in \partial\Omega; \end{cases}$$

it is discretized by means of the well-known 5-point finite difference formula to obtain the matrix $\mathbf{A} = \mathbf{I}_N \otimes \mathbf{H}_N + \mathbf{H}_N \otimes \mathbf{I}_N$. The associated function, given by $a(x, y) = 4 - 2\cos(x) - 2\cos(y)$, is nonnegative with a zero of multiplicity two in (0, 0); because of the zero, the condition number of $\mathbf{A}$ grows as N^2.

An adequate projector is therefore given by

$$\mathbf{p} = (\mathbf{C}_2^{(0)} \otimes \mathbf{I}_M + \mathbf{I}_N \otimes \mathbf{C}_1^{(\pi)}) \times (\mathbf{C}_1^{(\pi)} \otimes \mathbf{I}_M + \mathbf{I}_N \otimes \mathbf{C}_2^{(0)}), \tag{15}$$

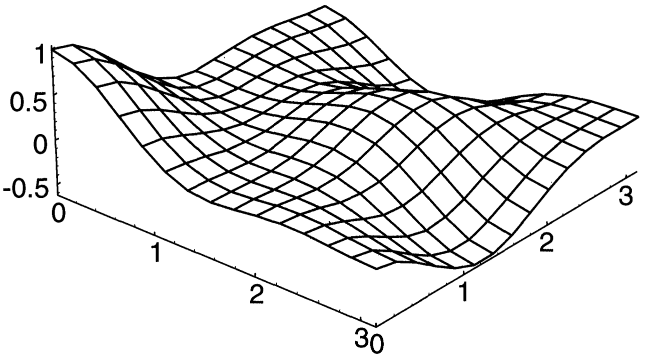


FIG. 2. *Scaled $p(x, y)$ as defined in* (15).

TABLE 1
Two-dimensional Laplacian operator.

Iter.	Pre=2 Post=0	Pre=1 Post=1
2	1.717 e+00	1.123 e+00
4	3.206 e−01	8.150 e−02
6	8.085 e−02	9.543 e−03
8	2.242 e−02	1.217 e−03
10	6.488 e−03	1.600 e−04
12	1.919 e−03	2.134 e−05
14	5.750 e−04	2.871 e−06
16	1.736 e−04	3.885 e−07
18	5.275 e−05	5.277 e−08
20	1.609 e−05	7.188 e−09

where

$$\mathbf{C}_1^{(\pi)} = \begin{bmatrix} 2 & 1 & & & O \\ 1 & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ O & & & 1 & 2 \end{bmatrix} \quad \text{and} \quad \mathbf{C}_2^{(0)} = \begin{bmatrix} -2 & 0 & 1 & & O \\ 0 & \ddots & \ddots & \ddots & \\ 1 & \ddots & \ddots & \ddots & 1 \\ & \ddots & \ddots & \ddots & 0 \\ O & & 1 & 0 & -2 \end{bmatrix}.$$

As required, $\mathbf{p}$ is such that its generating function $p(x, y)$ vanishes at the mirror points $(0, \pi)$, $(\pi, 0)$, and (π, π) as shown in Fig. 2.

The smoothers have been constructed as in (13) and (14) using $\alpha = 8$, i.e., the maximum of $a(x, y)$ on $Q = [0, \pi]^2$.

The results of the first 20 iterations reported in Table 1 show that in both cases the convergence rate is linear.

Note that, keeping the same computational cost, we achieve a considerable improvement using both the pre- and postsmoother to reduce the error on complementary subspaces.

6.3. The square function. In order to test the method under the same conditions of ill conditioning but in a full matrix case, we consider the matrix $\mathbf{A}$ generated by the function

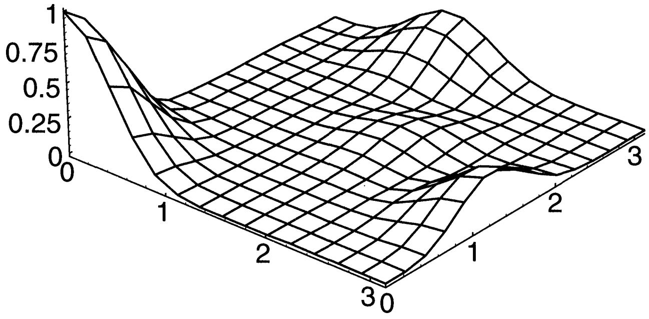


FIG. 3. *Scaled $p(x, y)$ as defined in* (16).

TABLE 2
The square function.

Iter.	Pre=2 Post=0	Pre=1 Post=1
2	2.903 e+00	2.108 e+00
4	1.016 e+00	4.586 e−01
6	4.314 e−01	1.387 e−01
8	2.007 e−01	4.635 e−02
10	9.807 e−02	1.626 e−02
12	4.933 e−02	5.869 e−03
14	2.531 e−02	2.155 e−03
16	1.317 e−02	7.998 e−04
18	6.926 e−03	2.989 e−04
20	3.668 e−03	1.123 e−04

$a(x, y) = x^2+y^2$. The condition number of this matrix grows quadratically because $a(0, 0) = 0$ with multiplicity two.

The matrix $\mathbf{A}$ can be written as $\mathbf{Q}_N \otimes \mathbf{I}_N + \mathbf{I}_N \otimes \mathbf{Q}_N$, where $\mathbf{Q}_N$ is generated by the trigonometric series for $f(x) = x^2$:

$$x^2 = \frac{\pi^2}{3} + 4\sum_{i=1}^{+\infty} \frac{(-1)^i \cos(ix)}{i^2}.$$

This time we use a projector better suited to correct the flat zero in (0, 0):

$$\mathbf{p} = [(\mathbf{C}_2^{(0)} \otimes \mathbf{I}_M + \mathbf{I}_N \otimes \mathbf{C}_1^{(\pi)}) \times (\mathbf{C}_1^{(\pi)} \otimes \mathbf{I}_M + \mathbf{I}_N \otimes \mathbf{C}_2^{(0)})]^2. \tag{16}$$

See Fig. 3.

The smoothers have been constructed from (13) and (14) using $a = 18$, an approximation of $\max_Q a(x, y)$.

The result of 20 iterations, reported Table 2, shows the effectiveness of the method also in the case of full matrices.

6.4. The absolute value. A more critical case comes from the matrix associated with the function $a(x, y) = |x| + |y|$. The matrix $\mathbf{A}_{N,M}$ can be constructed as in the previous case from the series expansion

$$|x| = \frac{\pi}{2} - \frac{4}{\pi}\sum_{i=0}^{+\infty} \frac{\cos((2i + 1)x)}{(2i + 1)^2}.$$

TABLE 3
The absolute value.

Iter.	Pre=2 Post=0	Pre=1 Post=1
2	1.770 e+00	1.075 e+00
4	3.249 e−01	7.321 e−02
6	7.840 e−02	7.058 e−03
8	2.013 e−02	7.064 e−04
10	5.292 e−03	7.159 e−05
12	1.406 e−03	7.294 e−06
14	3.763 e−04	7.452 e−07
16	1.010 e−04	7.626 e−08
18	2.719 e−05	7.813 e−09
20	7.329 e−06	8.012 e−10

TABLE 4
Comparison of the methods.

	Preconditioned	Multigrid	
Iter.	conj. grad.	Pre=2 Post=0	Pre=1 Post=1
2	5.304 e−01	1.933 e+00	1.127 e+00
4	5.162 e−02	2.900 e−01	1.015 e−02
6	7.175 e−03	5.069 e−02	1.189 e−04
8	8.065 e−04	9.212 e−03	1.468 e−06
10	9.608 e−05	1.699 e−03	1.859 e−08
12	9.731 e−06	3.159 e−04	2.399 e−10
14	9.384 e−07	5.897 e−05	3.151 e−12
16	9.152 e−08	1.103 e−05	4.212 e−14
18	1.191 e−08	2.069 e−06	5.731 e−16
20	1.337 e−09	3.883 e−07	7.929 e−18

Observe that, for any fixed value of N and M, the corresponding matrix $\mathbf{A}_{N,M}$ is generated by the truncated bivariate trigonometric series of $a(x, y)$. Actually, this polynomial is regular in its domain, whereas the limit function $a(x, y)$ is not differentiable at (0, 0). Using the full function as a guideline to construct a projector may seem to be a poor approximation; on the contrary, using projector (15) again and smoothers obtained with $\alpha = 2\pi$, we find again a convergence rate independent of the dimension. See Table 3.

6.5. Comparison of the methods. As a final example we report a comparison between the preconditioned conjugated gradient as proposed [S2] and our method on a case treated in [KK]; the matrix arises from the discretisation of a partial differential equation [CC] by means of a finite difference scheme and is constructed as follows.

Denoting with $\mathbf{BST}_N[a_1, \ldots, a_k]$ the $N \times N$ band symmetric Toeplitz matrix with $a_1, \ldots, a_k$ as the first entries in the first row, we may write the matrix of the problem as

$$\mathbf{A} = \mathbf{I} \otimes \mathbf{B} + \mathbf{H} \otimes \mathbf{C} + \mathbf{K} \otimes \mathbf{D},$$

where

$$\mathbf{B} = \mathbf{BST}_N[1, -0.12, -0.04], \qquad \mathbf{C} = \mathbf{BST}_N[-0.12, -0.04, -0.02],$$
$$\mathbf{D} = \mathbf{BST}_N[-0.04, -0.02, -0.01], \quad \mathbf{H} = \mathbf{BST}_N[0 \quad 1], \quad \text{and} \quad \mathbf{K} = \mathbf{BST}_N[0 \quad 0 \quad 1].$$

Since $a(0, 0) = 0$ with multiplicity two, we calibrate our multigrid method using projector (15) and smoothers (13) and (14) with $\alpha = 1.25$, i.e., an approximation for $\max_Q a(x, y)$.

In Table 4 we compare the error reductions of the preconditioned conjugate gradients of [S2] and our multigrid scheme.

Note that the preconditioners proposed in [CC, KK] do not apply to this case because the resulting condition number is $O(N)$. The convergence rate of these methods is sublinear requiring $O(N)$ steps to reduce the error by a constant factor.

REFERENCES

[BC] D. Bini and M. Capovani, *Spectral and computational properties of band symmetric matrices*, Linear Algebra Appl., 52/53 (1983), pp. 99–126.

[BP] D. Bini and V. Pan, *Matrix and polynomial computations*, in Fundamental Algorithms, Vol. 1, Birkhäuser-Verlag, Basel, Switzerland, 1994.

[CC] R. H. Chan and T. F. Chan, *Circulant preconditioners for elliptic problems*, J. Numer. Linear Algebra Appl., 1 (1992), pp. 77–101.

[CJ] R. H. Chan and X. Q. Jin, *A family of block preconditioners for block systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 1218–1235.

[Da] P. Davis, *Circulant Matrices*, J. Wiley and Sons, New York, 1979.

[DB] F. Di Benedetto, *Preconditioning of block Toeplitz matrices by sine transforms*, SIAM J. Sci. Comput., 17 (1996).

[FS] G. Fiorentino and S. Serra, *Multigrid Methods for Toeplitz Matrices*, Calcolo, 28 (1991), pp. 283–305.

[Gr] A. Greenbaum, *Analysis of a multigrid method as an iterative technique for solving linear systems*, SIAM J. Numer. Anal., 21 (1984), pp. 473–485.

[GZ] U. Grenander and G. Szego, *Toeplitz Forms and Their Applications*, 2nd ed., Chelsea, New York, 1984.

[H1] W. Hackbusch, *Multigrid Methods and Applications*, Springer-Verlag, Berlin, 1985.

[H2] ———, *Multigrid Methods* II, in Lecture Notes in Mathematics, Vol. 1228, Springer-Verlag, Berlin, 1987.

[KK] T. K. Ku and C. C. J. Kuo, *On the spectrum of a family of preconditioned Toeplitz matrices*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 948–966.

[Mu] B. Musicus, *Levinson and Fast Cholesky Algorithms for Toeplitz and almost Toeplitz Matrices*, Tech. report, Research Laboratory of Electronics, MIT, Cambridge, MA, 1981.

[Ri] J. Rissanen, *Algorithms for triangular decomposition of block Hankel and Toeplitz matrices with applications to factoring positive matrix polynomials*, Math. Comp., 27 (1973), pp. 147–154.

[S1] S. Serra, *Multi-iterative Methods*, Comput. Math. Appl., 26 (1993), pp. 65–87.

[S2] ———, *Preconditioning techniques for asymptotically ill-conditioned block Toeplitz systems with nonnegative generating functions*, BIT, 34 (1994), pp. 579–594.

[Tr] F. Trench, *An algorithm for the inversion of finite Toeplitz matrices*, J. Soc. Indust. Appl. Math., 12 (1964), pp. 515–522.

SIAM J. Sci. Comput.
Vol. 17, No. 5, pp. 1082–1091, September 1996

DEFECT CORRECTION FOR CONVECTION-DOMINATED FLOW*

WILHELM HEINRICHS†

Abstract. A defect correction scheme with the first-order upwind preconditioner is considered. By Fourier analysis the preconditioning properties for the second-order upwind scheme, the central scheme, and spectral methods are examined. Since the eigenvalues of the preconditioned operator are complex the generalized minimal residual iteration is used for the iterative solution. This procedure is applied to the Boussinesq flow problem in vorticity–streamfunction formulation. Numerical results are presented for increasing Rayleigh numbers.

Key words. convection–diffusion, defect correction, finite differences, spectral, multigrid, Boussinesq flow

AMS subject classification. 65N35

1. Introduction.

Here we consider convection–diffusion problems which can in their most general form be written as

$$-\epsilon \Delta u + c u_x + d u_y = f \ \text{ in } \Omega = (-1, 1)^2, \tag{1.1}$$

$$u = g \ \text{ on } \partial\Omega, \tag{1.2}$$

where $\epsilon > 0$ denotes a constant, c, d, f are functions defined in Ω, and g is defined on $\partial\Omega$. Such problems arise after a linearization of the Navier–Stokes equations (or Boussinesq flow problem). Here ϵ corresponds to $\frac{1}{Re}$. The part $-\epsilon\Delta u$ denotes the diffusive part and $cu_x + du_y$ denotes the convective part of the above equation. Here we are mainly interested in convection-dominated flows where $\epsilon << h$. Here h denotes the step size of the finite difference (FD) scheme. For the FD approximation of convection–diffusion problems one observes certain phenomena of instability. For small ϵ standard discretizations can lead to a solution of the discrete problem which has nothing to do with the solution of the original problem. For instance, the discretization of

$$-\epsilon u'' - 2u' = 0 \ \text{ in } [0, \infty),$$
$$u(0) = 1, \ u(\infty) = 0$$

with central differences yields as the discrete solution

$$u_{\epsilon,h}(ih) = \left(\frac{\epsilon - h}{\epsilon + h}\right)^i .$$

This is an $O(h^2)$ discretization. For ih fixed and $\frac{h}{\epsilon} \to 0$ we obtain

$$|(u_{\epsilon,h} - u)(ih)| = \left| \left(\frac{\epsilon - h}{\epsilon + h}\right)^i - \left(e^{-\frac{2h}{\epsilon}}\right)^i \right| \leq C \left(\frac{h}{\epsilon}\right)^2$$

with C independent of i, h, and ϵ. But the solution of the reduced equation is

$$u_{0,h}(ih) = \lim_{\epsilon \to 0} u_{\epsilon,h}(ih) = (-1)^i,$$

which means that for $\epsilon << h$, the solution of the FD problem has nothing to do with the exact solution

$$u(x) = e^{-\frac{2x}{\epsilon}}.$$

*Received by the editors February 1, 1993; accepted for publication (in revised form) May 2, 1995.

†Mathematisches Institut der Heinrich-Heine-Universität Düsseldorf, Universitätsstraße 1, D-4000 Düsseldorf 1, Germany (wheinric@numerik.uni-duesseldorf.de).

For this special problem we further observe a boundary layer in which the first derivative behaves as $O(\epsilon^{-1})$ for $\epsilon \to 0$. One possibility to avoid the phenomenon of instability is to use upstream discretization for u'. Clearly, an obvious disadvantage of this scheme lies in the fact that the method now becomes only first-order accurate. Hence it makes sense to use the first-order upstream scheme only as a preconditioner for a higher-order scheme. We analyze the preconditioning properties of this method for the following higher-order schemes:

- second-order upstream scheme,
- central finite difference scheme,
- Chebyshev pseudospectral scheme.

In the spectral scheme (see [4]) the solution is approximated by Chebyshev polynomials of degree $\leq N$. This space is denoted by $\mathbf{P}_N$. By a Fourier analysis it can be shown that the eigenvalues of the preconditioned operator are bounded but complex. Hence one has to employ a nonsymmetric matrix iteration for the solution. Here we recommend the generalized minimal residual (GMRES) iteration which belongs to the residual minimization methods.

Clearly, for the general convection–diffusion problem (1.1), (1.2) the first derivatives u_x and u_y have to be approximated according to the sign of the coefficients c and d, respectively. Therefore for the iterative solution we recommend flow directed schemes. Since the Chebyshev nodes are dense near the boundary, it is necessary to use line Gauss–Seidel relaxation (in an alternating manner). Finally this iterative solver is applied to the Boussinesq flow problem in vorticity–streamfunction formulation. We obtained numerical results for increasing Rayleigh numbers up to $Ra = 10^5$.

2. Preconditioning by the upstream scheme. From the one-dimensional model problem it can be seen that for $\epsilon \to 0$ we first have to find a good preconditioner for the derivative operator

$$\frac{du}{dx}.$$

Here we employ the first-order upstream scheme (L^1_{up}) for preconditioning, i.e.,

$$\frac{du}{dx}(x_i) \cong \frac{u(x_{i+1}) - u(x_i)}{x_{i+1} - x_i} \quad \text{if } x_{i+1} < x_i$$

or

$$\frac{du}{dx}(x_i) \cong \frac{u(x_i) - u(x_{i-1})}{x_i - x_{i-1}} \quad \text{if } x_{i-1} < x_i.$$

In the spectral discretization we have collocation points $x_i = \cos\theta_i$, $\theta_i = \frac{i\pi}{N}$, so that $x_{i+1} < x_i$, whereas in the finite difference case we have grid points $x_i = ih$, $h = \frac{1}{N}$, so that $x_{i-1} < x_i$ for $i = 1, \dots, N-1$. Now we were interested in the preconditioning properties of L^1_{up} for the following three higher-order methods:

- second-order upstream scheme: $L^2_{up}u := \frac{1}{h}\left(\frac{1}{2}u(x_{i-2}) - 2u(x_{i-1}) + \frac{3}{2}u(x_i)\right)$,
- second-order central scheme: $L_{ce}u := \frac{1}{2h}\left(u(x_{i+1}) - u(x_{i-1})\right)$,
- Chebyshev pseudospectral scheme: L_{sp} (see [3], [4], [12], [13]).

Hence we are interested in the eigenvalues of the discrete operators

$$\left(L^1_{up}\right)^{-1} L^2_{up}, \ \left(L^1_{up}\right)^{-1} L_{ce}, \ \left(L^1_{up}\right)^{-1} L_{sp}.$$

Eigenvalue bounds are obtained by a Fourier analysis. It is well known (see [4]) that the Fourier analysis also yields a good prediction for the eigenvalues in the Chebyshev case. For

TABLE 1
ρ_k *for* $k = 1, \ldots, 4$.

k	ρ_k
1	0.3333
2	0.2425
3	0.2162
4	0.2040

$\left(L^1_{up}\right)^{-1} L^2_{up}$ the absolute values of the eigenvalues are given by

$$|\lambda^{up}_p| = \sqrt{4 - 3\cos^2 \frac{p\pi}{2N}}, \quad p = 1, \ldots, N-1.$$

For $\left(L^1_{up}\right)^{-1} L_{ce}$ they are given by

$$|\lambda^{ce}_p| = \cos \frac{p\pi}{2N}, \quad p = 1, \ldots, N-1.$$

Since $\cos \frac{p\pi}{2N} \in [0, 1]$ we obtain

$$|\lambda^{up}_p| \in [1, 2] \text{ and } |\lambda^{ce}_p| \in [0, 1].$$

Because zero is the lower bound for the eigenvalues of the preconditioned central scheme, it is already clear that this method is not good. For the second-order upstream scheme we observed that the imaginary parts are small compared to the real parts. Hence a simple Richardson iteration can be applied. By choosing one relaxation parameter we obtain a convergence factor of $\frac{1}{3}$. Clearly, the convergence speed can be accelerated by using more parameters (nonstationary Richardson relaxation [11]). As shown in [11] the convergence factor ρ_k for k relaxations is here given by

$$\rho_k := |T_k(3)|^{-\frac{1}{k}},$$

where T_k denotes the kth Chebyshev polynomial. In Table 1 we present ρ_k for $k = 1, \ldots, 4$. We recommend using $k = 3$ since the improvement for increasing k is no more significant.

For the spectral Fourier operator (see [4, §5.2.2]) the eigenvalues of the preconditioned spectral operator are

$$\lambda^{sp}_p = \frac{p\pi/N}{\sin(p\pi/N)} e^{-i\frac{p\pi}{N}}, \quad p = -\frac{N}{2}, \ldots, \frac{N}{2},$$

and hence

$$|\lambda^{sp}_p| = \frac{p\pi/N}{\sin(p\pi/N)}, \quad p = -\frac{N}{2}, \ldots, \frac{N}{2}.$$

This implies that

$$|\lambda^{sp}_p| \in \left[1, \frac{\pi}{2}\right] \cong [1, 1.57].$$

Therefore in all three cases the eigenvalues are bounded but complex. The iterative solver must be able to handle complex eigenvalues. Here we recommend the GMRES iteration (see [19]–[21]) which belongs to the residual minimization methods. Consider the general linear system

$$Bv = g,$$

where B is a large nonsymmetric matrix. If v_0 is an initial approximation to the solution and $r_0 = g - Bv_0$, we define the mth Krylov subspace

$$K_m := \text{span}\{r_0, Br_0, B^2r_0, \ldots, B^{m-1}r_0\}.$$

Then the GMRES approximation v_m with

$$v_m \in v_0 + K_m \tag{2.1}$$

is determined such that the mth residual r_m fulfills

$$||r_m||_2 = \text{minimum}. \tag{2.2}$$

An equivalent statement is the orthogonality condition

$$r_m \perp BK_m. \tag{2.3}$$

The GMRES iteration is a robust implementation of (2.1)–(2.3) by means of an Arnoldi construction of an orthonormal basis for the Krylov space, which leads to an $(m+1) \times m$ Hessenberg least-squares solution [21].

3. Stabilization techniques and iterative solvers. Here we consider convection–diffusion problems which are in their most general form given by (1.1), (1.2). It is well known that these problems lead to instabilities for $\epsilon << h$ in the FD case (see [7], [8], [10]) and for $\epsilon << N^{-2}$ in the spectral case (see [2], [15]). Canuto [2] has shown that the spectral approximations are affected by spurious oscillations which deteriorate the spectral accuracy. For instance, for the one-dimensional model problem

$$-\epsilon u'' + u' = 0 \ \text{ in } (-1, 1), \tag{3.1}$$

$$u(-1) = 0, \ u(1) = 1, \tag{3.2}$$

the exact solution is given by

$$u_\epsilon(x) = \frac{e^{\frac{x+1}{\epsilon}} - 1}{e^{\frac{2}{\epsilon}} - 1}. \tag{3.3}$$

Hence the boundary layer exhibited near $x = 1$ when $\epsilon \to 0$ has a width of order $O(\epsilon)$.

The pseudospectral approximation $u_N \in \mathbf{P}_N$ of (3.1), (3.2) is now defined by

$$-\epsilon u_N''(x_i) + u_N'(x_i) = 0 \ \text{ for } i = 1, \ldots, N-1, \tag{3.4}$$

$$u_N(-1) = 0, \ u_N(1) = 1, \tag{3.5}$$

where $x_i = \cos\frac{i\pi}{N}$ denotes the Chebyshev collocation points. In [2] the spectral approximation is explicitly calculated and finally one obtains

$$\begin{aligned} u_N &\cong \tfrac{1}{2} + \tfrac{1}{2}T_N \quad \text{for odd } N, \\ u_N &\cong \hat{u}_0 + \hat{u}_N T_N, |\hat{u}_0| \cong \hat{u}_N \cong O(\epsilon N^2)^{-1} \quad \text{for even } N \end{aligned}$$

as $\epsilon \to 0$, $\epsilon << N^{-2}$. Here T_N denotes the Nth Chebyshev polynomial.

Therefore in both cases u_N is strongly oscillating but for a given ϵ the oscillations created by the boundary layer are less pronounced if N is chosen to be odd. This shows that attention should be paid to the parity of the degree of polynomials to be used in a spectral approximation of boundary layer problems. This example further demonstrates the instability of the convection-dominated problem. In particular, for even N we can read from the coefficients $\hat{u}_0$ and $\hat{u}_N$ that the approximation error is perturbed by an instability rate of $O(\epsilon N^2)^{-1}$.

The problem of instability is also well known for finite difference (or finite element) discretizations if central finite differences are used (see, e.g., [9]). Here also spurious oscillations are introduced by the discretization scheme. This problem is avoided by applying an upstream scheme where the first derivative is approximated by a one-sided finite difference star. Another possibility is to add an artificial diffusion (or viscosity) term of the form $-N^{-1}\Delta u$ to the convection–diffusion equation. Now these methods are stable and produce nonoscillating solutions but are only first-order accurate being based on solving a modified problem. Clearly such techniques are of no interest for spectral approximations since the high spectral accuracy is completely lost.

Therefore we thought of other techniques of stabilization which maintain the high accuracy of spectral methods. Together with Eisen [8] we obtained a stable scheme by adding one additional equation of collocation to the original system. Hence we obtain an overdetermined system of equations and the instability caused by the highest mode is avoided. This is a certain kind of penalty method. We prove stability independent of ϵ and present the numerically calculated condition numbers for several types of collocation points (Chebyshev Gauss or Gauss–Lobatto nodes). A drawback of this approach is that the method is not flow directed and therefore in our model problem for $\epsilon \to 0$, $\epsilon << N^{-2}$, the spectral solution approximates the straight line $u_s(x) = 0.5(x+1)$ (see [8, Fig. 2]) instead of the boundary layer solution (3.3). Another drawback is that for overdetermined systems no efficient iterative solvers like multigrid methods are available. For these reasons this method is probably not the best way to go.

A flow directed method is the *streamline diffusion method* which was introduced by Hughes and Brooks [17], [18] for a finite element discretization. This method is stable and no accuracy is lost. It is a Petrov–Galerkin modification of the standard Galerkin method where artificial diffusion in the streamline direction is introduced by modifying the test functions from v to

$$v + \delta_x c v_x + \delta_y d v_y,$$

where $\delta_x = \delta_y = O(h)$ and h denotes the step size of the finite element scheme. Clearly, for the spectral method with Chebyshev Gauss–Lobatto points one has to choose a point-dependent viscosity given by $\delta_x = \delta_x(x)$, $\delta_y = \delta_y(y)$, where

$$\delta_x(x_i) = C_x \sin\left(\frac{\pi}{N}\right) \sin\left(i\frac{\pi}{N}\right), \quad i = 1, \ldots, N-1,$$
$$\delta_y(y_j) = C_y \sin\left(\frac{\pi}{N}\right) \sin\left(j\frac{\pi}{N}\right), \quad j = 1, \ldots, N-1$$

with suitable constants C_x, C_y. These formulas result from the finite difference discretization with central differences (see [11]). The constants C_x, C_y can be chosen such that the resulting finite difference matrix yields an M-matrix (see [9]), i.e., its inverse has only nonnegative entries. For the practically more efficient pseudospectral method, the stabilization is achieved by adding the viscosity term to the right-hand side f. Here f is replaced by

$$f + \delta_x c f_x + \delta_y d f_y,$$

and the corresponding differential operator is modified such that the new problem is equivalent to the original system (1.1), (1.2). In [15] we investigated this method in connection with a multidomain approach for the above boundary layer problem. Stability is shown and suitable multigrid components for the efficient solution of the stabilized problem are presented. Clearly, a drawback of this approach is that high-order derivatives have to be computed, which makes it quite expensive.

For preconditioning of the original spectral system we recommend an upwind FD method, for which each of the first derivatives u_x and u_y is differenced according to the sign of the coefficients c and d, respectively. The Laplace operator is discretized by standard central finite differences and the first derivatives u_x and u_y in (x_i, y_j), $i, j = 1, \dots, N-1$ are approximated as follows:

$$\begin{aligned}
c(x_i, y_j) \geq 0 : \quad & u_x(x_i, y_j) \cong \frac{u(x_{i+1}, y_j) - u(x_i, y_j)}{x_{i+1} - x_i}, \\
c(x_i, y_j) < 0 : \quad & u_x(x_i, y_j) \cong \frac{u(x_i, y_j) - u(x_{i-1}, y_j)}{x_i - x_{i-1}}, \\
d(x_i, y_j) \geq 0 : \quad & u_y(x_i, y_j) \cong \frac{u(x_i, y_{j+1}) - u(x_i, y_j)}{y_{j+1} - y_j}, \\
d(x_i, y_j) < 0 : \quad & u_y(x_i, y_j) \cong \frac{u(x_i, y_j) - u(x_i, y_{j-1})}{y_j - y_{j-1}}.
\end{aligned}$$

For the iterative solution we recommend flow directed schemes. Since the Chebyshev nodes are dense near the boundary it is necessary to use line Gauss–Seidel relaxation. For smoothing it is recommended to use alternate iterations of flow directed horizontal iterations (FDHI) and flow directed vertical iterations (FDVI). In the literature this combination is called FDHVI (see [7], [10]). The iterative scheme FDHI is a variant of line Gauss–Seidel relaxation. Let P_i denote the mesh points on the vertical line $x = x_i$. We divide P_i into two subsets:

$$\begin{aligned}
P_{i,E} &:= \{(i, j) : c(x_i, y_j) \geq 0\}, \\
P_{i,W} &:= \{(i, j) : c(x_i, y_j) < 0\}.
\end{aligned}$$

The FDHI partitioning and ordering of the unknowns consists of the subsets $P_{i,E}$ arranged in order of increasing i, followed by the subsets $P_{i,W}$, arranged in order of decreasing i. The difference equations on each of the subsets $P_{i,E}$ or $P_{i,W}$ are a collection of tridiagonal systems. By considering the mesh points P_j on a horizontal line $y = y_j$ and dividing P_j into subsets $P_{j,N}$, $P_{j,S}$, we may construct the iterative scheme FDVI. Finally one alternates between FDHI and FDVI, resulting in FDHVI. For a more detailed description including numerical results, we refer the reader to [7] and [10]. Han et al. [10] describe a procedure based on directed graphs to partition and order the unknowns of the Gauss–Seidel process. This is performed by inspection of the coefficient matrix. Nevertheless, this algorithm is expensive for nonlinear problems, like those coming from the Navier–Stokes or Boussinesq equations, when the coefficients are solution dependent and require the reconstruction of the directed graph several times. The penalty for such a choice is proportional to the number of mesh points. In §4 the FDHVI scheme is applied to the Boussinesq flow problem.

4. Application to the Boussinesq flow problem. The problem specifically considered here is that of the two-dimensional flow of a Boussinesq fluid of Prandtl number $Pr = 0.71$ (i.e., air) in an upright square cavity (see [1], [6]). The walls are nonslip and impermeable. The horizontal walls are adiabatic and the vertical sides are at fixed temperatures. In addition to the Navier–Stokes equations we have one further equation for the temperature T. By Ra we denote the Rayleigh number. The Boussinesq flow problem in vorticity–streamfunction formulation reads as follows:

$$4\Delta\psi + \omega = 0 \text{ in } \Omega = (-1,1)^2, \tag{4.1}$$

$$-2Pr\Delta\omega + \frac{\partial}{\partial x}(v_1\omega) + \frac{\partial}{\partial y}(v_2\omega) = RaPr\frac{\partial T}{\partial x} \text{ in } \Omega, \tag{4.2}$$

$$-2\Delta T + \frac{\partial}{\partial x}(v_1 T) + \frac{\partial}{\partial y}(v_2 T) = 0 \text{ in } \Omega. \tag{4.3}$$

As usual $(v_1, v_2)^t$ denotes the velocity. The scalar factors 2 in equations (4.2) and (4.3) and 4 in equation (4.1) are due to the fact that we here define the problem in $(-1,1)^2$ instead of the original square cavity $(0,1)^2$. ψ fulfills homogeneous Dirichlet boundary conditions, i.e.,

$$\psi = \frac{\partial\psi}{\partial\nu} = 0 \text{ on } \partial\Omega$$

and T fulfills mixed Dirichlet–Neumann boundary conditions, i.e.,

$$T(-1,y) = 1,\ T(1,y) = 0 \text{ for } y \in (-1,1),$$
$$\frac{\partial T}{\partial y}(x,-1) = \frac{\partial T}{\partial y}(x,1) = 0 \text{ for } x \in [-1,1].$$

The homogeneous Neumann boundary conditions correspond to the fact that the horizontal walls are adiabatic.

Now the equations (4.1)–(4.3) are linearized by a quasi-Newton method (see [16]), where the velocity from the previous iteration is employed. Hence we have to solve the linear problem:

$$4\Delta\psi^{n+1} + \omega^{n+1} = 0 \text{ in } \Omega, \tag{4.4}$$

$$-2Pr\Delta\omega^{n+1} + \frac{\partial}{\partial x}(v_1^n\omega^{n+1}) + \frac{\partial}{\partial y}(v_2^n\omega^{n+1}) = RaPr\frac{\partial T^{n+1}}{\partial x} \text{ in } \Omega, \tag{4.5}$$

$$-2\Delta T^{n+1} + \frac{\partial}{\partial x}(v_1^n T^{n+1}) + \frac{\partial}{\partial y}(v_2^n T^{n+1}) = 0 \text{ in } \Omega, \tag{4.6}$$

where the index n denotes the nth iterate.

Clearly, the velocity is related to the streamfunction by

$$v_1^n = \frac{\partial\psi^n}{\partial y},\quad v_2^n = -\frac{\partial\psi^n}{\partial x}.$$

The linear system (4.4)–(4.6) is now approximately solved by a spectral multigrid (SMG) method (see [12], [13]). We first describe the pseudospectral discretization of the system (4.4)–(4.6). The functions ψ^{n+1} and ω^{n+1} are spectrally approximated by polynomials $u_{N+2} \in \mathbf{P}_{N+2}$, $v_N \in \mathbf{P}_N$. T^{n+1} is approximated by a polynomial $w_N \in \mathbf{P}_N$. Here the index $n+1$ is omitted. Hence v_1^n, v_2^n are approximated by the polynomials $v_{1,N}$, $v_{2,N} \in \mathbf{P}_{N+2}$, where

$$v_{1,N} = \frac{\partial\bar{u}_{N+2}}{\partial y},\quad v_{2,N} = -\frac{\partial\bar{u}_{N+2}}{\partial x},$$

and $\bar{u}_{N+2} \in \mathbf{P}_{N+2}$ corresponds to the polynomial u_{N+2} from the previous iteration. Now the pseudospectral problem related to (4.4)–(4.6) reads as follows: Find $u_{N+2} \in \mathbf{P}_{N+2}$, $v_N \in \mathbf{P}_N$, $w_N \in \mathbf{P}_N$ such that

$$[4\Delta u_{N+2} + v_N](x_i, y_j) = 0 \tag{4.7}$$

for $i, j = 0, \ldots, N$ and

$$\left[-2Pr\Delta v_N + \frac{\partial}{\partial x}(v_{1,N}v_N) + \frac{\partial}{\partial y}(v_{2,N}v_N)\right](x_i, y_j) = RaPr\frac{\partial w_N}{\partial x}(x_i, y_j), \tag{4.8}$$

$$\left[-2\Delta w_N + \frac{\partial}{\partial x}(v_{1,N}w_N) + \frac{\partial}{\partial y}(v_{2,N}w_N)\right](x_i, y_j) = 0 \tag{4.9}$$

for $i, j = 1, \ldots, N-1$. Since u has to fulfill two types of boundary conditions we choose $u_{N+2} \in \mathbf{P}_{N+2}$, fulfilling the homogeneous Dirichlet boundary conditions. For v_N there are no boundary conditions. The pseudospectral boundary conditions for $w_N \in \mathbf{P}_N$ are given by

$$w_N(-1, y_j) = 1, \; w_N(1, y_j) = 0 \;\text{ for } j = 1, \ldots, N-1, \tag{4.10}$$

$$\frac{\partial w_N}{\partial y}(x_i, -1) = \frac{\partial w_N}{\partial y}(x_i, 1) = 0 \;\text{ for } i = 0, \ldots, N. \tag{4.11}$$

The system (4.7)–(4.11) completely determines the spectral approximations u_{N+2}, v_N, w_N. It is clear that the polynomials u_{N+2}, v_N are uniquely determined by the equations (4.7), (4.8). Furthermore the equation (4.9) together with the boundary conditions (4.10), (4.11) uniquely determine the polynomial w_N. Hence in the linearized version the systems (4.7), (4.8) for determining u_{N+2}, v_N and (4.9)–(4.11) for determining w_N can be handled separately. First, one solves the system (4.9)–(4.11) for w_N by an SMG method, then one calculates $\frac{\partial w_N}{\partial x}(x_i, y_j)$, $i, j = 1, \ldots, N-1$, and finally one solves the system (4.7), (4.8) by the SMG method introduced in [14]. Here we employed six V-cycles of SMG to get a nearly exact solution of the linear systems.

Now we turn to a more precise description of the SMG method. We use the same components as already introduced in §3. A somewhat different treatment results from the fact that the diffusive part is now perturbed by the first-order derivatives $\frac{\partial}{\partial x}$, $\frac{\partial}{\partial y}$. For an increasing Rayleigh number the convective part becomes dominant. Hence in the defect correction step one has to use an FD approximation which remains stable also for an increasing Rayleigh number. Furthermore the FD problem has to be solved approximately by a suitable iterative method which also works for convection-dominated flows. Here we employed the FDHVI iteration for preconditioning of the spectral system resulting from the equations (4.8), (4.9). To handle the complex eigenvalues of the preconditioned spectral operator we employ nonsymmetric matrix iterations. Here we choose the GMRES iteration. For a more detailed description of these components, we refer to §3.

By using these components we numerically calculated for various Rayleigh numbers and mesh sizes the following quantities:

$|\psi|_{\text{mid}}$: absolute value of the streamfunction at the midpoint of the cavity,
$|\psi|_{\max}$: maximum absolute value of the streamfunction,
$v_{1,\max}$: maximum horizontal velocity on the vertical midplane of the cavity,
$v_{2,\max}$: maximum horizontal velocity on the horizontal midplane of the cavity.

The local heat flux in a horizontal direction at any point in the cavity is given by

$$Q := v_1 T - 2\frac{\partial T}{\partial x}.$$

Let us further introduce the following Nusselt numbers:

$\overline{Nu} := \frac{1}{4}\int_{-1}^{1}\int_{-1}^{1} Q(x, y)dxdy$: average Nusselt number throughout the cavity,
$Nu_{\frac{1}{2}} := \frac{1}{2}\int_{-1}^{1} Q(0, y)dy$: average Nusselt number on the vertical midplane,

TABLE 2
Results for $Ra = 10^3$.

N	$\lvert\psi\rvert_{\mathrm{mid}}$	$\lvert\psi\rvert_{\max}$	$v_{1,\max}$	$v_{2,\max}$	$\overline{Nu}$	$Nu_{\frac{1}{2}}$	Nu_0	$Nu_{\max}$	$Nu_{\min}$
8	1.1747	1.1747	3.5250	3.6122	1.1178	1.1173	1.1174	1.5048	0.6902
16	1.1746	1.1746	3.5690	3.6148	1.1178	1.1178	1.1178	1.5060	0.6913
24	1.1746	1.1746	3.6441	3.6759	1.1178	1.1178	1.1178	1.5060	0.6913

TABLE 3
Results for $Ra = 10^4$.

N	$\lvert\psi\rvert_{\mathrm{mid}}$	$\lvert\psi\rvert_{\max}$	$v_{1,\max}$	$v_{2,\max}$	$\overline{Nu}$	$Nu_{\frac{1}{2}}$	Nu_0	$Nu_{\max}$	$Nu_{\min}$
8	5.0713	5.0713	15.8009	18.9305	2.2474	2.1946	2.1870	3.6170	0.5067
16	5.0736	5.0736	15.8352	19.0474	2.2448	2.1946	2.1870	3.5314	0.5853
24	5.0981	5.0980	16.1800	19.5000	2.2340	2.2350	2.2420	3.5450	0.5920

TABLE 4
Results for $Ra = 10^5$.

N	$\lvert\psi\rvert_{\mathrm{mid}}$	$\lvert\psi\rvert_{\max}$	$v_{1,\max}$	$v_{2,\max}$	$\overline{Nu}$	$Nu_{\frac{1}{2}}$	Nu_0	$Nu_{\max}$	$Nu_{\min}$
8	14.3409	18.8519	37.8844	40.2643	4.4140	4.7345	4.7590	10.4740	0.3438
16	11.3720	12.3330	36.3420	61.3420	4.5030	4.5061	4.5313	7.9010	0.7551
24	9.1600	9.6530	34.6320	67.9120	4.5100	4.5120	4.5231	7.7700	0.7361

$Nu_0 := \frac{1}{2}\int_{-1}^{1} Q(-1, y)dy$: average Nusselt number on the vertical boundary,
$Nu_{\max} := \max\{|Q(-1, y)| : y \in [-1, 1]\}$: maximum value of the Nusselt number,
$Nu_{\min} := \min\{|Q(-1, y)| : y \in [-1, 1]\}$: minimum value of the Nusselt number.

The above integrals in the definition of $\overline{Nu}$, $Nu_{\frac{1}{2}}$, and Nu_0 are evaluated by the Clenshaw–Curtis quadrature (see [5, p. 68]). In Tables 2–4 we present the numerical results for different Rayleigh numbers and $N = 8, 16, 24$. The numerical results are in good accordance with the results obtained in [6]. However, for a larger Rayleigh number or increasing N the above SMG method is no more convergent. The reason is that upstream preconditioning is not good enough. Here one has to find some better ways of preconditioning. We are currently trying to find improved preconditioners where the finite difference discretization is performed on staggered grids.

REFERENCES

[1] M. BEHNIA, M. WOLFSTEIN, AND G. DE VAHL DAVIS, *A stable fast marching scheme for computational fluid mechanics*, Internat. J. Numer. Methods Fluids, 10 (1990) pp. 607–621.

[2] C. CANUTO, *Spectral methods and a maximum principle*, Math. Comp., 51 (1988) pp. 615–629.

[3] ———, *Parallelism in Spectral Methods*, I.A.N.-C.N.R. Report 609, Pavia, Italy, 1988.

[4] C. CANUTO, M. Y. HUSSAINI, A. QUARTERONI, AND T. A. ZANG, *Spectral Methods in Fluid Dynamics*, Springer Series in Computational Physics, Springer-Verlag, Berlin, Heidelberg, New York, 1989.

[5] P. DAVIS AND P. RABINOWITZ, *Methods of Numerical Integration*, Academic Press, Orlando, FL, 1984.

[6] G. DE VAHL DAVIS, *Natural convection of air in a square cavity: A bench mark numerical solution*, Internat. J. Numer. Methods Fluids, 3 (1983) pp. 249–264.

[7] M. D. DEVILLE AND E. H. MUND, *Finite element preconditioning of collocation schemes for advection-diffusion equations*, Proceedings of the IMACS International Symposium on Iterative Methods in the Linear Algebra, R. Beauwens and P. de Groen, eds., Brüssel, 1992, Elsevier, Amsterdam, 1992, pp. 181–189.

[8] H. EISEN AND W. HEINRICHS, *A new method of stabilization for singular perturbation problems with spectral methods*, SIAM J. Numer. Anal, 29 (1992), pp. 107–122.

[9] W. HACKBUSCH, *Theorie und Numerik elliptischer Differentialgleichungen*, Teubner Studienbücher, B. G. Teubner, Stuttgart, 1986.

[10] H. HAN, V. P. IL'IN, AND R. B. KELLOGG, *Flow directed iterations for convection dominated flow*, in BAIL V, Proceedings of the Fifth Internat. Conf. on Boundary and Interior Layers—Computational and Asymptotic Methods, G. Ben-yu, J. J. H. Miller, and S. Zhong-ci, eds., Shanghai, China, 1988.

[11] W. HEINRICHS, *Kollokationsverfahren und Mehrgittermethoden bei elliptischen Randwertaufgaben*, GMD-Bericht Nr. 168, Oldenbourg-Verlag, München-Wien, 1987.

[12] ———, *Line relaxation for spectral multigrid methods*, J. Comput. Phys., 77 (1988), pp. 166–182.

[13] ———, *Multigrid methods for combined finite difference and Fourier problems*, J. Comput. Phys., 78 (1988), pp. 424–436.

[14] ———, *Spectral multigrid techniques for the Stokes problem in streamfunction formulation*, J. Comput. Phys., 102 (1992), pp. 310–318.

[15] ———, *A stabilized multi domain approach for singular perturbation problems*, J. Sci. Comput., 7 (1992), pp. 95–125.

[16] ———, *Spectral projective Newton methods for quasilinear elliptic boundary value problems*, Calcolo, 29 (1993), pp. 33–48.

[17] T. J. R. HUGHES AND A. BROOKS, *A multidimensional upwind scheme with no crosswind diffusion*, in Finite Element Methods for Convection Dominated Flows, T. J. R. Hughes, ed., AMD Vol. 34, American Society of Mechanical Engineers, New York, 1979.

[18] ———, *A theoretical framework for Petrov-Galerkin methods with discontinuous weighting functions: Application to the streamline-upwind procedure*, in Finite Elements in Fluids, Vol. 4, R. H. Gallagher, ed., Wiley, New York, 1979.

[19] N. M. NACHTIGAL, S. C. REDDY, AND L. N. TREFETHEN, *How Fast Are Nonsymmetric Matrix Iterations*?, Numerical Analysis Report 90-2, Department of Mathematics, Massachusetts Institute of Technology, Cambridge, MA, 1990.

[20] Y. SAAD AND M. H. SCHULTZ, *Conjugate gradient-like algorithms for solving nonsymmetric linear systems*, Math. Comp., 44 (1985), pp. 417–424.

[21] ———, GMRES: *A generalized minimum residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

SIAM J. SCI. COMPUT.
Vol. 17, No. 5, pp. 1092–1121, September 1996

005

A MODEL NUMERICAL SCHEME FOR THE PROPAGATION OF PHASE TRANSITIONS IN SOLIDS*

BERNARDO COCKBURN† AND HUIING GAU‡

Abstract. In this paper, we devise a simple finite difference scheme that produces approximations to the viscosity-capillarity solutions of the equations that govern the propagation of phase transitions in solids (or to the equations of van der Waals fluids) for all positive values of the dimensionless parameter that characterizes the viscosity-capillarity solution. Numerical experiments showing the convergence properties of the method are presented.

Key words. viscosity-capillarity, phase transitions, mixed-type conservation laws

AMS subject classifications. 35M10, 35L65, 65L12, 65M12, 73V15

1. Introduction. In this paper, we devise and validate a method for numerically solving the initial value problem associated with the equations that govern the propagation of phase transitions in solids, namely,

$$
\begin{aligned}
\gamma_t &= v_x, \\
v_t &= (\sigma(\gamma))_x, \\
\gamma(0) &= \gamma_0, \\
v(0) &= v_0,
\end{aligned}
\tag{1.1}
$$

where v is the velocity, γ the strain, and σ the stress. We consider Lipschitz stresses that are increasing on phase 1 ($-1 < \gamma < \gamma_{1,2}$) and phase 3 ($\gamma_{2,3} < \gamma$) of the material and are decreasing on phase 2 ($\gamma_{1,2} < \gamma < \gamma_{2,3}$); see Fig. 1.

This renders (1.1) a system of mixed type, since it is well known that when $\sigma' > 0$ the system (1.1) is hyperbolic and that when $\sigma' < 0$ the system is elliptic. As pointed out by James in [6], this fact enriches the problem of finding a solution for (1.1) because now the standard entropy condition for hyperbolic systems is not enough to single out a physically relevant solution. A way to do so is through the so-called viscosity-capillarity (VC) approach which consists of obtaining solutions of (1.1) as limits of the solutions of the system

$$
\begin{aligned}
\tilde{\gamma}_t &= \tilde{v}_x, \\
\tilde{v}_t &= (\sigma(\tilde{\gamma}))_x + \nu\, \tilde{v}_{xx} - \lambda\, \tilde{\gamma}_{xxx}, \\
\tilde{\gamma}(0) &= \gamma_0, \\
\tilde{v}(0) &= v_0,
\end{aligned}
\tag{1.2}
$$

when the parameters ν and λ go to zero while the number $\omega = 2\sqrt{\lambda}/\nu$ is maintained fixed. The solutions of (1.1) obtained in this way will thus be called the VC solutions of (1.1). In this paper, we restrict ourselves to the problem of devising a numerical scheme capable of converging to those solutions.

Truskinovsky [14] and Slemrod [10] proposed independently the notion of VC solutions for the equations of van der Waals fluids, which are of the form (1.1). In this context, $\gamma < \gamma_{1,2}$ represents the liquid phase and $\gamma > \gamma_{2,3}$ the vapor phase; the phase $\gamma_{1,2} < \gamma < \gamma_{2,3}$ is the

*Received by the editors April 29, 1994; accepted for publication (in revised form) March 22, 1995.

†School of Mathematics, University of Minnesota, 127 Vincent Hall, Minneapolis, MN 55455 (cockburn@math.umn.edu). The research of this author was partially supported by NSF grant DMS-91-03997 and by a grant from the Minnesota Supercomputer Institute.

‡School of Mathematics, University of Minnesota, 127 Vincent Hall, Minneapolis, MN 55455 (gau@math.umn.edu).

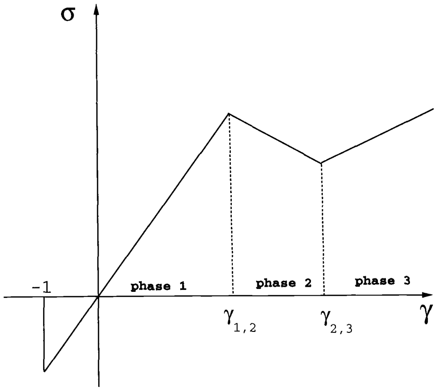


FIG. 1. *Example of strain–stress relation.*

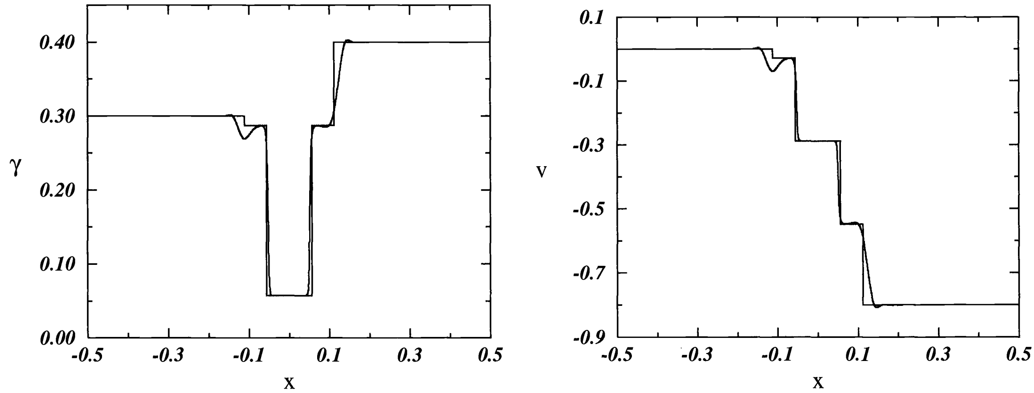


FIG. 2. *Exact VC solution and its approximation for* $\omega = 3$.

region in which the system is elliptic. The VC criterion re-incorporates in equations (1.1) the physics lost in neglecting the second-order and third-order terms of (1.2). In [5], a review of recent results about the VC solutions of the Riemann problem for (1.1) can be found.

There are very few papers in the available literature concerning numerical methods for mixed-type systems like (1.1). In [9], Slemrod proposed the idea of using the Lax–Friedrichs scheme to numerically approximate the VC solution of (1.1) for $\omega = 1$. This idea was later successfully numerically tested in the pioneering work of Slemrod and Flaherty [11] who considered a Lax–Friedrichs-type scheme with which they could handle the case $\omega \in [0, 1]$. In [12], Shu developed a technique that allows the use of the essentially nonoscillatory schemes for numerically solving mixed-type problems. In [7], Jin considered formally second-order accurate relaxation Lax–Friedrichs-type schemes for VC solutions with $\omega \in [0, 1]$. Our simple finite difference scheme is the first scheme to work for all nonnegative values of ω. Affouf and Caflish [4] used a second-order finite difference scheme to study the stability of the solutions of Riemann problems for (1.2); they took the parameters ν and λ to be order-one parameters. Pitman and Ni [8] have also studied mixed systems like (1.1) with a so-called visco-elastic relaxation law.

To give an idea of the performance of our model scheme, we display in Fig. 2 the exact VC solution of (1.1) for $\omega = 3$ and the strain–stress curve of Fig. 1 with the following parameter values:

$$\gamma_{1,2} = .1, \ \gamma_{2,3} = .2, \ \sigma'(\gamma) = \begin{cases} 20, & \gamma \in [-1, \gamma_{1,2}), \\ -10, & \gamma \in (\gamma_{1,2}, \gamma_{2,3}), \\ 5, & \gamma \in (\gamma_{2,3}, \infty). \end{cases}$$

Thus, in this case the elliptic region (phase 2) is the region $\gamma \in [.1, .2]$. The initial data is

$$\gamma_0(x) = \begin{cases} .3, & x < 0, \\ .4, & x > 0, \end{cases} \qquad v_0(x) = \begin{cases} 0, & x < 0, \\ -.8, & x > 0. \end{cases}$$

The exact VC strain displays two 1–3 phase transitions (or boundaries) near the origin and two shocks both of which occur within phase 3. Note that the phase boundaries are very well approximated and that no spurious oscillations or spikes are present.

We stress the fact that in this paper we are mainly concerned with the problem of how to deal with the presence of the elliptic region. The quality of the approximation outside the elliptic region can be improved by using several well-known techniques for hyperbolic conservation laws. This will be done in a forthcoming paper.

There are several other criteria to select a solution for systems like (1.1); see, for example, the papers by Truskinovsky [15] and Abeyaratne and Knowles [1], and the survey by Truskinovsky [16]. We consider this paper to be the first step in our effort of devising a numerical scheme capable of converging to the solutions singled out by general criteria.

The paper is organized as follows. In §2, we present the heuristics that will guide us in devising our numerical scheme which we construct in §3. In §4, we study the stability of its semidiscrete version and obtain information about the effect of the discretization parameters on the stability of the scheme. In §5, we display our numerical results which indicate that the scheme converges to the VC solutions of (1.1) for a wide range of values of the dimensionless parameter ω. We end in §6 with some concluding remarks.

2. Heuristics. As in [11], the idea that guides us in devising our numerical scheme is very simple: the numerical scheme must have a model equation whose solutions behave like the solutions of (1.2).

We look for numerical schemes whose model equation is of the following form:

$$\begin{aligned} \hat{\gamma}_t &= \hat{v}_x + r_0 \nu_0 \hat{\gamma}_{xx}, \\ \hat{v}_t &= (\sigma(\hat{\gamma}))_x + \nu_0 \hat{v}_{xx} - \lambda_0 \hat{\gamma}_{xxx}. \end{aligned} \tag{2.1a}$$

This form of the model equation is very convenient because by changing variables as follows:

$$\bar{\gamma} = \hat{\gamma}, \tag{2.2a}$$

$$\bar{v} = \hat{v} + r_0 \nu_0 \hat{\gamma}_x, \tag{2.2b}$$

we can rewrite system (2.1) as

$$\bar{\gamma}_t = \bar{v}_x, \tag{2.3a}$$

$$\bar{v}_t = (\sigma(\bar{\gamma}))_x + \nu_0 (1 + r_0) \bar{v}_{xx} - (\lambda_0 + r_0 \nu_0^2) \bar{\gamma}_{xxx}, \tag{2.3b}$$

and if we set

$$\omega = 2 \frac{\sqrt{\lambda_0 + r_0 \nu_0^2}}{(1 + r_0) \nu_0}, \tag{2.4}$$

we see that the functions $(\bar{v}, \bar{\gamma})$ given by (2.2), (2.1a) and satisfying the initial conditions

$$\bar{\gamma}(0) = \gamma_0, \qquad \bar{v}(0) = v_0, \tag{2.1b}$$

are exactly equal to $(\tilde{v}, \tilde{\gamma})$ for $\nu = (1 + r_0)\nu_0$. Thus, to obtain the scheme we are looking for, we simply have to discretize (2.1) and (2.2) in a suitable way.

In [11], Slemrod and Flaherty took the viscosity coefficient ν_0 proportional to $(\Delta x)^{2/3}$ and $\lambda_0 = 0$. The relation (2.4) becomes then

$$\omega = 2\frac{\sqrt{r_0}}{(1 + r_0)},$$

which implies that ω must lie in the interval $[0, 1]$. By (2.4), to allow the parameter ω to be bigger than one, λ_0 must be strictly bigger than zero. In this paper, we explore the following choices of the discretization parameters:

$$r_0 = 1, \tag{2.5a}$$

$$\nu_0 = \left\{ \sup_{\gamma \geq -1} \max\{0, \sigma'(\gamma)\}\right\}^{1/2} (\Delta x)^\beta, \tag{2.5b}$$

$$\lambda_0 = (\omega^2 - 1)\,\nu_0^2, \tag{2.5c}$$

for several values of β in $(0, 1]$. Note that (2.5c) follows from (2.5a) and (2.4).

If we pick the very convenient choice $\beta = 1$, in order to obtain a numerical scheme with (2.1a) as a model equation, we simply have to discretize (2.1a) with

(1) a third-order accurate approximation of the first-order space derivatives,
(2) a second-order accurate approximation of the second-order space derivatives,
(3) a consistent approximation of the third-order derivative,
(4) a third-order accurate time discretization.

In what follows, we construct such a scheme.

Note that only the parameters $\nu = (1 + r_0)\nu_0$ and $\lambda = \lambda_0 + r_0\,\nu_0^2$ have an effect on the properties of the solution $(\bar{\gamma}, \bar{v})$ of (2.1) and (2.2). However, the particular value of the parameter r_0, for example, may have an influence on the properties of the approximate solution due to the properties of the discretization procedure. In §4, we show that this is indeed the case for the scheme we devise next. We have found, experimentally, that the choice (2.5a) works well.

3. A finite difference scheme. For the sake of simplicity, we consider uniform partitions in both space and time and set, as usual, $x_j = j\,\Delta x$, $x_{j+1/2} = (j + 1/2)\Delta x$, $t^n = n\Delta t$. We denote by u_j^n an approximation of $u(t^n, x_j)$ and by $u_{j+1/2}^n$ an approximation of $u(t^n, x_{j+1/2})$. We assume that $n \in \mathbb{N}$ and that $j \in \mathbb{Z}$.

3.1. Discretizing equations (2.1a). The following space discretizations satisfy the above requirements:

$$(u_x)_j \sim (u_{j-2} - 8\,u_{j-1} + 8\,u_{j+1} - u_{j+2})/12\Delta x,$$
$$(u_{xx})_j \sim (u_{j-1} - 2\,u_j + u_{j+1})/(\Delta x)^2,$$
$$(u_{xxx})_j \sim (-u_{j-2} + 2\,u_{j-1} - 2\,u_{j+1} + u_{j+2})/2(\Delta x)^3.$$

We can rewrite the resulting space discretization of (2.1) in conservation form as follows:

$$\frac{d}{dt}U_j = \frac{1}{\Delta x}(F_{j+1/2} - F_{j-1/2}), \tag{3.1a}$$

where

$$U_j = \begin{pmatrix} \hat{\gamma}_j \\ \hat{v}_j \end{pmatrix}, \quad F_{j+1/2}(U) = \begin{pmatrix} (f_v)_{j+1/2} \\ (f_\sigma)_{j+1/2} \end{pmatrix}, \tag{3.1b}$$

and

$$\begin{aligned} (f_v)_{j+1/2} &= (-\hat{v}_{j-1} + 7\hat{v}_j + 7\hat{v}_{j+1} - \hat{v}_{j+2})/12 \\ &\quad + r_0\, \nu_0\, (-\hat{\gamma}_j + \hat{\gamma}_{j+1})/\Delta x, \end{aligned} \tag{3.1c}$$

$$\begin{aligned} (f_\sigma)_{j+1/2} &= \sigma((-\hat{\gamma}_{j-1} + 7\hat{\gamma}_j + 7\hat{\gamma}_{j+1} - \hat{\gamma}_{j+2})/12) \\ &\quad + \nu_0\, (-\hat{v}_j + \hat{v}_{j+1})/\Delta x \\ &\quad - \lambda_0\, (\hat{\gamma}_{j-1} - \hat{\gamma}_j - \hat{\gamma}_{j+1} + \hat{\gamma}_{j+2})/2(\Delta x)^2. \end{aligned} \tag{3.1d}$$

Finally, we discretize in time by the following third-order accurate TVD-Runge–Kutta time stepping [13]:

$$\begin{aligned} V_j &= U_j^n + \frac{\Delta t}{\Delta x}(F_{j+1/2}(U^n) - F_{j-1/2}(U^n)), \\ W_j &= \frac{3}{4}\, U_j^n + \frac{1}{4}\left\{ V_j^n + \frac{\Delta t}{\Delta x}(F_{j+1/2}(V) - F_{j-1/2}(V))\right\}, \\ U_j^{n+1} &= \frac{1}{3}\, U_j^n + \frac{2}{3}\left\{ W_j^n + \frac{\Delta t}{\Delta x}(F_{j+1/2}(W) - F_{j-1/2}(W))\right\}. \end{aligned} \tag{3.2}$$

3.2. The initial condition and the definition of the approximate solution. Note that a discrete version of (2.2) which is consistent with our space discretization (3.1a), (3.1c) is the following:

$$\begin{aligned} \bar{\gamma}_j &= \hat{\gamma}_j, \\ \bar{v}_{j+1/2} &= (f_v)_{j+1/2}. \end{aligned} \tag{3.3}$$

With this way of defining the approximate strain and velocity, the spike that appears in the approximate velocity at the phase boundaries reported by [11] is totally eliminated; see §5.

To obtain the initial values $(\hat{\gamma}_j^0, \hat{v}_j^0)$, we must take into account (3.3). To do so, we proceed as follow. First, we discretize the initial condition (2.1b) as usual; namely,

$$\begin{aligned} \bar{\gamma}_j^0 &= \frac{1}{\Delta x}\int_{x_{j-1/2}}^{x_{j+1/2}} \gamma_0(x)\, dx, \\ \bar{v}_{j+1/2}^0 &= \frac{1}{\Delta x}\int_{x_j}^{x_{j+1}} v_0(x)\, dx. \end{aligned} \tag{3.4}$$

Then, we solve the linear system set

$$\begin{aligned} &\hat{\gamma}_j^0 = \bar{\gamma}_j^0, \\ &(-\hat{v}_{j-1}^0 + 7\hat{v}_j^0 + 7\hat{v}_{j+1}^0 - \hat{v}_{j+2}^0)/12 = s_{j+1/2}, \end{aligned} \tag{3.5}$$

where

$$s_{j+1/2} = \bar{v}_{j+1/2}^0 - r_0\, \nu_0(-\bar{\gamma}_j^0 + \bar{\gamma}_{j+1}^0)/\Delta x. \tag{3.6a}$$

However, to avoid having to invert the matrix equation $My = b$ given by (3.5) (which could render the initial value $\hat{v}^0$ a highly oscillating function), we consider the matrix equation $M^t Mx = M^t b$ and we *lump the masses* of the matrix $M^t M$. We thus obtain the following simple approximate evaluation of the initial condition:

$$\hat{\gamma}_j^0 = \bar{\gamma}_j^0, \tag{3.6b}$$

$$\hat{v}_j^0 = (-s_{j-3/2}^0 + 7s_{j-1/2}^0 + 7s_{j+1/2}^0 - s_{j+3/2}^0)/12. \tag{3.6c}$$

3.3. Definition of the scheme. We are now ready to define the numerical scheme. Given any $\omega \geq 0$, we compute an approximation of the corresponding VC solution of (1.1) at time T as follows:

(1) compute the initial data $(\bar{v}^0, \bar{\gamma}^0)$ by using (3.4);
(2) compute $(\hat{v}^0, \hat{\gamma}^0)$ by using (3.6);
(3) set $\Delta t = T/N$ for some N suitably chosen (see §5.1);
(4) for $n = 1$ to $N-1$, compute $(\hat{v}^{n+1}, \hat{\gamma}^{n+1})$ by using the scheme (3.2), (3.1);
(5) compute the wanted approximation $(\bar{v}^N, \bar{\gamma}^N)$ by using (3.3).

4. Stability analysis of the semidiscrete scheme. We now turn to the question of the stability of our numerical scheme. Our goal is to prove that the semidiscrete version of the scheme is stable. To do so, we mimic the procedure for obtaining stability for the solutions of our model equation which we show next for the sake of completeness.

Multiplying equation (2.3b) by $\bar{v}$ and integrating over x, we get, after integrating by parts,

$$\frac{1}{2}\frac{d}{dt}\int_{\mathbb{R}} \bar{v}^2 = -\int_{\mathbb{R}} \sigma(\bar{\gamma})\,\bar{v}_x - \nu_0(1+r_0)\int_{\mathbb{R}}(\bar{v}_x)^2 + (\lambda_0 + r_0\nu_0^2)\int_{\mathbb{R}} \bar{\gamma}_{xx}\,\bar{v}_x.$$

By (2.3a), $\bar{v}_x = \bar{\gamma}_t$, and so we get

$$\begin{aligned}\frac{1}{2}\frac{d}{dt}\int_{\mathbb{R}} \bar{v}^2 &= -\int_{\mathbb{R}} \sigma(\bar{\gamma})\,\bar{\gamma}_t - \nu_0(1+r_0)\int_{\mathbb{R}}(\bar{\gamma}_t)^2 + (\lambda_0 + r_0\nu_0^2)\int_{\mathbb{R}} \bar{\gamma}_{xx}\,\bar{\gamma}_t \\ &= -\frac{d}{dt}\int_{\mathbb{R}} W(\bar{\gamma}) - \nu_0(1+r_0)\int_{\mathbb{R}}(\bar{\gamma}_t)^2 - \frac{1}{2}(\lambda_0 + r_0\nu_0^2)\frac{d}{dt}\int_{\mathbb{R}}(\bar{\gamma}_x)^2,\end{aligned}$$

where $W' = \sigma$. Thus,

$$\frac{d}{dt}E(\bar{v}, \bar{\gamma}) = -\nu_0(1+r_0)\int_{\mathbb{R}}(\bar{\gamma}_t)^2, \tag{4.1a}$$

where the energy $E(\bar{v}, \bar{\gamma})$ is defined as follows:

$$E(\bar{v}, \bar{\gamma}) = \frac{1}{2}\int_{\mathbb{R}} \bar{v}^2 + \int_{\mathbb{R}} W(\bar{\gamma}) + \frac{1}{2}(\lambda_0 + r_0\nu_0^2)\int_{\mathbb{R}}(\bar{\gamma}_x)^2 \tag{4.1b}$$

or, if the choice (2.4) for λ_0 is taken,

$$E(\bar{v}, \bar{\gamma}) = \frac{1}{2}\int_{\mathbb{R}} \bar{v}^2 + \int_{\mathbb{R}} W(\bar{\gamma}) + \frac{1}{2}\left(\frac{1+r_0}{2}\right)^2 \nu_0^2\,\omega^2\int_{\mathbb{R}}(\bar{\gamma}_x)^2. \tag{4.1c}$$

We now follow the above process to try to obtain a similar stability result for the semidiscrete scheme (3.1). We proceed in several steps.

4.1. Rewriting the scheme. We start by rewriting the semidiscrete scheme in terms of $\bar{v}$ and $\bar{\gamma}$. To facilitate such a rewriting, we introduce the shifting operators S_i defined by $S_i u_k = u_{k+i}$. Setting $M = (-S_{-1} + 7S_0 + 7S_1 - S_2)/12$ and $D = (S_{-1} - S_0 - S_1 + S_2)/2$, we can now rewrite the semidiscrete scheme (3.1) as follows:

$$\frac{d}{dt}\hat{\gamma}_j = (-S_{-1} + S_0)(M\hat{v}_j + r_0\nu_0(-S_0 + S_1)\hat{\gamma}_j/\Delta x)/\Delta x,$$
$$\frac{d}{dt}\hat{v}_j = (-S_{-1} + S_0)(\sigma(M\hat{\gamma}_j) + \nu_0(-S_0 + S_1)\hat{v}_j/\Delta x - \lambda_0 D\hat{\gamma}_j/(\Delta x)^2)/\Delta x.$$

Since, by (3.3), we have

$$\begin{aligned}\bar{\gamma}_j &= \hat{\gamma}_j,\\ \bar{v}_{j+1/2} &= M\hat{v}_j + r_0\nu_0(-S_0 + S_1)\bar{\gamma}_j/\Delta x,\end{aligned} \tag{4.2}$$

we can write the first equation of the above scheme as follows:

$$\frac{d}{dt}\bar{\gamma}_j = (-S_{-1} + S_0)\bar{v}_{j+1/2}/\Delta x. \tag{4.3a}$$

To express the second equation in terms of $\bar{\gamma}$ and $\bar{v}$, we proceed as follows. By (4.2), we have

$$\begin{aligned}\frac{d}{dt}\bar{v}_{j+1/2} &= M\frac{d}{dt}\hat{v}_j + r_0\nu_0(-S_0 + S_1)\frac{d}{dt}\bar{\gamma}_j\\ &= M\frac{d}{dt}\hat{v}_j + r_0\nu_0(-S_0 + S_1)(-S_{-1} + S_0)\bar{v}_{j+1/2}/\Delta x,\end{aligned}$$

by (4.3a). Inserting the expression for $\frac{d}{dt}\hat{v}_j$ and rearranging terms, we obtain

$$\begin{aligned}\frac{d}{dt}\bar{v}_{j+1/2} = {}& M(-S_{-1} + S_0)\sigma(M\hat{\gamma}_j)/\Delta x\\ &+ \nu_0(-S_{-1} + S_0)(-S_0 + S_1)(M\hat{v}_j + r_0\bar{v}_{j+1/2})/(\Delta x)^2\\ &- \lambda_0(-S_{-1} + S_0)\, M\, D\hat{\gamma}_j/(\Delta x)^3.\end{aligned}$$

Finally, since, by (4.2), $M\hat{v}_j = \bar{v}_{j+1/2} - r_0\nu_0(-S_0 + S_1)\bar{\gamma}_j/\Delta x$, we have

$$\begin{aligned}\frac{d}{dt}\bar{v}_{j+1/2} = {}& M(-S_{-1} + S_0)\sigma(M\hat{\gamma}_j)/\Delta x\\ &+ (1 + r_0)\nu_0(-S_0 + S_1)(-S_{-1} + S_0)\bar{v}_{j+1/2}/(\Delta x)^2\\ &- r_0\nu_0^2(-S_{-1} + S_0)(-S_0 + S_1)^2\bar{\gamma}_j/(\Delta x)^3\\ &- \lambda_0(-S_{-1} + S_0)\, M\, D\hat{\gamma}_j/(\Delta x)^3\\ \equiv {}& \Theta^1_{j+1/2} + \Theta^2_{j+1/2} + \Theta^3_{j+1/2} + \Theta^4_{j+1/2}.\end{aligned} \tag{4.3b}$$

4.2. Multiplying by $\bar{v}_{j+1/2}$. Following the procedure for the model equation, we multiply (4.3b) by $\bar{v}_{j+1/2}\,\Delta x$ and sum over j. We get

$$\frac{1}{2}\frac{d}{dt}\sum_{j\in\mathbb{Z}}(\bar{v}_{j+1/2})^2\,\Delta x = \sum_{j\in\mathbb{Z}}(\Theta^1_{j+1/2} + \Theta^2_{j+1/2} + \Theta^3_{j+1/2} + \Theta^4_{j+1/2})\bar{v}_{j+1/2}\,\Delta x.$$

Summing by parts, we obtain

$$\begin{aligned}\sum_{j\in\mathbb{Z}} \Theta^1_{j+1/2}\bar{v}_{j+1/2}\,\Delta x &= \sum_{j\in\mathbb{Z}}[M(-S_{-1}+S_0)\sigma(M\hat{\gamma}_j)]\,\bar{v}_{j+1/2}\\ &= \sum_{j\in\mathbb{Z}}[M\sigma(M\hat{\gamma}_j)]\,[(-S_1+S_0)\bar{v}_{j+1/2}]\\ &= -\sum_{j\in\mathbb{Z}}[M\sigma(M\hat{\gamma}_j)]\,\frac{d}{dt}\bar{\gamma}_{j+1}\Delta x\end{aligned}$$

by (4.3a). Summing by parts once more, we get

$$\sum_{j\in\mathbb{Z}} \Theta^1_{j+1/2}\bar{v}_{j+1/2}\,\Delta x = -\sum_{j\in\mathbb{Z}}\sigma(M\hat{\gamma}_j)\,\frac{d}{dt}[M\bar{\gamma}_j]\Delta x = -\frac{d}{dt}\sum_{j\in\mathbb{Z}}W(M\hat{\gamma}_j)\,\Delta x,$$

since $W' = \sigma$. For the second term, we have

$$\begin{aligned}\sum_{j\in\mathbb{Z}} \Theta^2_{j+1/2}\bar{v}_{j+1/2}\,\Delta x &= (1+r_0)\nu_0\sum_{j\in\mathbb{Z}}[(-S_0+S_1)(-S_{-1}+S_0)\bar{v}_{j+1/2}]\,\bar{v}_{j+1/2}/(\Delta x)\\ &= (1+r_0)\nu_0\sum_{j\in\mathbb{Z}}[(-S_{-1}+S_0)\bar{v}_{j+1/2}]\,[(-S_0+S_{-1})\bar{v}_{j+1/2}]/(\Delta x)\\ &= -(1+r_0)\nu_0\sum_{j\in\mathbb{Z}}\left(\frac{d}{dt}\bar{\gamma}_j\right)^2\,(\Delta x)\end{aligned}$$

by (4.3a). For the third term, after a summation by parts, we get

$$\begin{aligned}\sum_{j\in\mathbb{Z}} \Theta^3_{j+1/2}\bar{v}_{j+1/2}\,\Delta x &= -r_0\nu_0^2\sum_{j\in\mathbb{Z}}[(-S_{-1}+S_0)(-S_0+S_1)^2\bar{\gamma}_j]\,[\bar{v}_{j+1/2}]/(\Delta x)^2\\ &= -r_0\nu_0^2\sum_{j\in\mathbb{Z}}[(-S_{-1}+S_0)(-S_0+S_1)\bar{\gamma}_j]\\ &\qquad\times[(-S_0+S_{-1})\bar{v}_{j+1/2}]/(\Delta x)^2\\ &= r_0\nu_0^2\sum_{j\in\mathbb{Z}}[(-S_{-1}+S_0)(-S_0+S_1)\bar{\gamma}_j]\,\frac{d}{dt}\bar{\gamma}_j/\Delta x\end{aligned}$$

by (4.3a). Hence, summing by parts once more, we obtain

$$\begin{aligned}\sum_{j\in\mathbb{Z}} \Theta^3_{j+1/2}\bar{v}_{j+1/2}\,\Delta x &= r_0\nu_0^2\sum_{j\in\mathbb{Z}}[(-S_0+S_1)\bar{\gamma}_j]\,\frac{d}{dt}[(-S_1+S_0)\bar{\gamma}_j]/\Delta x\\ &= -\frac{1}{2}r_0\nu_0^2\frac{d}{dt}\sum_{j\in\mathbb{Z}}(-\bar{\gamma}_j+\bar{\gamma}_{j+1})^2/\Delta x.\end{aligned}$$

Finally, for the fourth term, we have

$$\begin{aligned}\sum_{j\in\mathbb{Z}} \Theta^4_{j+1/2}\bar{v}_{j+1/2}\,\Delta x &= -\lambda_0\sum_{j\in\mathbb{Z}}[(-S_{-1}+S_0)\,M\,D\hat{\gamma}_j]\,\hat{v}_{j+1/2}/(\Delta x)^2\\ &= -\lambda_0\sum_{j\in\mathbb{Z}}[M\,D\hat{\gamma}_j]\,[(-S_1+S_0)\,\hat{v}_{j+1/2}]/(\Delta x)^2\end{aligned}$$

$$= \lambda_0 \sum_{j\in\mathbb{Z}} [M\, D\hat{\gamma}_j] \left[\frac{d}{dt}\bar{\gamma}_{j+1}\right] /(\Delta x)$$

$$= \lambda_0 \sum_{j\in\mathbb{Z}} [D\hat{\gamma}_j] \left[M\frac{d}{dt}\bar{\gamma}_j\right] /(\Delta x).$$

Since

$$\begin{aligned} M\hat{\gamma}_j &= (-\hat{\gamma}_{j-1} + 7\hat{\gamma}_j + 7\hat{\gamma}_{j+1} - \hat{\gamma}_{j+2})/12 \\ &= (-\hat{\gamma}_{j-1} + \hat{\gamma}_j + \hat{\gamma}_{j+1} - \hat{\gamma}_{j+2})/12 + (\hat{\gamma}_j + \hat{\gamma}_{j+1})/2 \\ &= -D\hat{\gamma}_j/6 + (S_0 + S_1)\hat{\gamma}_j/2, \end{aligned}$$

we can write

$$\sum_{j\in\mathbb{Z}} \Theta^4_{j+1/2}\bar{v}_{j+1/2}\,\Delta x = -\frac{1}{48}\lambda_0\frac{d}{dt}\sum_{j\in\mathbb{Z}}(\bar{\gamma}_{j-1} - \bar{\gamma}_j - \bar{\gamma}_{j+1} + \bar{\gamma}_{j+2})^2/(\Delta x) + \Psi,$$

where

$$\Psi = \lambda_0\frac{1}{2}\sum_{j\in\mathbb{Z}}[D\hat{\gamma}_j]\left[(S_0 + S_1)\frac{d}{dt}\hat{\gamma}_j\right]/\Delta x.$$

Since $D = (S_0 - S_2)(S_{-1} - S_0)/2$, we have

$$\begin{aligned} \Psi &= \frac{1}{4}\lambda_0\sum_{j\in\mathbb{Z}}[(S_0 - S_2)\hat{\gamma}_j]\,\frac{d}{dt}[(S_1 - S_0)(S_0 + S_1)\hat{\gamma}_j]/\Delta x \\ &= \frac{1}{4}\lambda_0\sum_{j\in\mathbb{Z}}[(S_0 - S_2)\hat{\gamma}_j]\,\frac{d}{dt}[(-S_0 + S_2)\hat{\gamma}_j]/\Delta x \\ &= -\frac{1}{8}\lambda_0\frac{d}{dt}\sum_{j\in\mathbb{Z}}(-\hat{\gamma}_{j-1} + \hat{\gamma}_{j+1})^2/\Delta x. \end{aligned}$$

Putting together the above equalities, we find that we have the following discrete version of (4.1):

$$\frac{d}{dt}E_h = -\nu_0(1 + r_0)\sum_{j\in\mathbb{Z}}\left(\frac{d}{dt}\bar{\gamma}_j\right)^2\Delta x, \tag{4.4a}$$

where the approximate energy E_h is defined by

$$\begin{aligned} E_h = {} & \frac{1}{2}\sum_{j\in\mathbb{Z}}(\bar{v}_{j+1/2})^2\,\Delta x + \sum_{j\in\mathbb{Z}} W((-\bar{\gamma}_{j-1} + 7\bar{\gamma}_j + 7\bar{\gamma}_{j+1} - \bar{\gamma}_{j+2})/12)\,\Delta x \\ & + \frac{1}{2}r_0\nu_0^2\sum_{j\in\mathbb{Z}}(-\bar{\gamma}_j + \bar{\gamma}_{j+1})^2/\Delta x + \frac{1}{2}\lambda_0\sum_{j\in\mathbb{Z}}\frac{1}{4}(-\bar{\gamma}_{j-1} + \bar{\gamma}_{j+1})^2/\Delta x \\ & + \frac{1}{48}\lambda_0\sum_{j\in\mathbb{Z}}(\bar{\gamma}_{j-1} - \bar{\gamma}_j - \bar{\gamma}_{j+1} + \bar{\gamma}_{j+2})^2/\Delta x. \end{aligned} \tag{4.4b}$$

4.3. The energy E_h for the choice (2.4) of λ_0. Next, we find the expression of E_h for the choice (2.4) λ_0; namely,

$$\lambda_0 = \left(\frac{1+r_0}{2}\right)^2 \nu_0^2\,\omega^2 - r_0\,\nu_0^2.$$

Inserting this form of λ_0 in the expression of E_h (4.4b), we obtain

$$\begin{aligned} E_h &= \frac{1}{2}\sum_{j\in\mathbb{Z}}(\bar{v}_{j+1/2})^2\,\Delta x + \sum_{j\in\mathbb{Z}} W((-\bar{\gamma}_{j-1} + 7\bar{\gamma}_j + 7\bar{\gamma}_{j+1} - \bar{\gamma}_{j+2})/12)\,\Delta x \\ &+ \frac{1}{2}\left(\frac{1+r_0}{2}\right)^2 \nu_0^2\,\omega^2 \sum_{j\in\mathbb{Z}} \frac{1}{4}(-\bar{\gamma}_{j-1} + \bar{\gamma}_{j+1})^2/\Delta x \\ &+ \frac{1}{48}\left(\frac{1+r_0}{2}\right)^2 \nu_0^2\,\omega^2 \sum_{j\in\mathbb{Z}}(\bar{\gamma}_{j-1} - \bar{\gamma}_j - \bar{\gamma}_{j+1} + \bar{\gamma}_{j+2})^2/\Delta x + r_0\,\nu_0^2\,R_{0h}/\Delta x, \end{aligned}$$

where

$$R_{0h} = \sum_{j\in\mathbb{Z}}\left\{\frac{1}{2}(-\bar{\gamma}_j + \bar{\gamma}_{j+1})^2 - \frac{1}{48}(\bar{\gamma}_{j-1} - \bar{\gamma}_j - \bar{\gamma}_{j+1} + \bar{\gamma}_{j+2})^2 - \frac{1}{8}\,(-\bar{\gamma}_{j-1} + \bar{\gamma}_{j+1})^2\right\}.$$

Setting $z_{j+1/2} = -\bar{\gamma}_j + \bar{\gamma}_{j+1}$ and $\rho_j = -z_{j-1/2} + z_{j+1/2}$, we rewrite R_{0h} as follows:

$$\begin{aligned} R_{0h} &= \sum_{j\in\mathbb{Z}}\left\{\frac{1}{2}\,(z_{j+1/2})^2 - \frac{1}{48}(-z_{j-1/2} + z_{j+3/2})^2 - \frac{1}{8}\,(z_{j-1/2} + z_{j+1/2})^2\right\} \\ &= \frac{1}{48}\sum_{j\in\mathbb{Z}}\left\{24\,(z_{j+1/2})^2 - (-z_{j-1/2} + z_{j+3/2})^2 - 6\,(z_{j-1/2} + z_{j+1/2})^2\right\} \\ &= \frac{1}{48}\sum_{j\in\mathbb{Z}}\left\{6\,(-z_{j-1/2} + z_{j+1/2})^2 - (-z_{j-1/2} + z_{j+3/2})^2\right\} \\ &= \frac{1}{48}\sum_{j\in\mathbb{Z}}\left\{6\,(\rho_j)^2 - (\rho_j + \rho_{j+1})^2\right\} \\ &= \frac{1}{48}\sum_{j\in\mathbb{Z}}\left\{2(\rho_j)^2 + (-\rho_j + \rho_{j+1})^2\right\} \end{aligned}$$

or

$$R_{0h} = \frac{1}{48}\sum_{j\in\mathbb{Z}}\left\{2(\bar{\gamma}_{j-1} - 2\bar{\gamma}_j + \bar{\gamma}_{j+1})^2 + (-\bar{\gamma}_{j-1} + 3\bar{\gamma}_j - 3\bar{\gamma}_{j+1} + \bar{\gamma}_{j+2})^2\right\}.$$

As a consequence, we can write that with the choice (2.4) the energy E_h is indeed a nonnegative quantity given by

$$
\begin{aligned}
E_h = {} & \frac{1}{2}\sum_{j\in\mathbb{Z}} (\bar{v}_{j+1/2})^2\,\Delta x + \sum_{j\in\mathbb{Z}} W((-\bar{\gamma}_{j-1} + 7\bar{\gamma}_j + 7\bar{\gamma}_{j+1} - \bar{\gamma}_{j+2})/12)\,\Delta x \\
& + \frac{1}{2}\left(\frac{1+r_0}{2}\right)^2 \nu_0^2\,\omega^2 \sum_{j\in\mathbb{Z}} \frac{1}{4}(-\bar{\gamma}_{j-1} + \bar{\gamma}_{j+1})^2/\Delta x \\
& + \frac{1}{48}\left(\frac{1+r_0}{2}\right)^2 \nu_0^2\,\omega^2 \sum_{j\in\mathbb{Z}} (\bar{\gamma}_{j-1} - \bar{\gamma}_j - \bar{\gamma}_{j+1} + \bar{\gamma}_{j+2})^2/\Delta x \\
& + \frac{1}{48} r_0\,\nu_0^2 \sum_{j\in\mathbb{Z}} \left\{ 2(\bar{\gamma}_{j-1} - 2\bar{\gamma}_j + \bar{\gamma}_{j+1})^2 + (-\bar{\gamma}_{j-1} + 3\bar{\gamma}_j - 3\bar{\gamma}_{j+1} + \bar{\gamma}_{j+2})^2 \right\}/\Delta x.
\end{aligned} \tag{4.4c}
$$

We summarize the results obtained thus far in this section in the following result.

PROPOSITION 4.1 (stability of the semidiscrete scheme (3.1)). *Let the energy E_h be the quantity defined by* (4.4b) *(or* (4.4c) *when λ_0 satisfies* (2.4)*). Then, the equality* (4.4a) *holds and so, for all $t \geq 0$, we have*

$$
E_h(t) + \nu_0(1 + r_0) \int_0^t \sum_{j\in\mathbb{Z}} \left(\frac{d}{dt}\bar{\gamma}_j\right)^2 \Delta x = E_h(0).
$$

In what follows, we argue that (i) the case $\omega \ll 1$ is more difficult to treat than the case $\omega \gg 1$, (ii) it is crucial to maintain the "stability enhancer" parameter r_0 strictly positive, and (iii) the artificial viscosity ν_0 must be reasonably large.

4.4. On the parameters ω, r_0, and ν_0. Notice that it is very important to ensure that the numerical scheme damps away spurious oscillations in the approximate strains. The presence of these oscillations might prevent the convergence to the correct VC solution of (1.1) because, since the stored energy W is nonlinear, even if the approximate strain converges weakly to γ, the approximate stored energy might not converge (weakly) to $W(\gamma)$. From the above stability result, it is clear that the role of the last three terms of the energy E_h is to control those oscillations.

The effect of the size of ω. From the expression (4.4c) of the energy and from the stability result of Proposition 4.1, it is clear that when $\omega \ll 1$ the control on the oscillations of the approximate strain is relaxed and that if $\omega \gg 1$ such control is enhanced.

The effect of the size of r_0. When $\omega = 0$, control on the possible spurious oscillations in the approximate strain can be exerted only if $r_0 > 0$. Moreover, even when $\omega > 0$ it is still crucial to take r_0 strictly positive. Notice that the third and fourth terms of the energy E_h are equal to zero for (and only for) the highly oscillatory wave

$$
\bar{\gamma}_j = \begin{cases} \bar{\gamma}_1 & \text{for } j \text{ even}, \\ \bar{\gamma}_2 & \text{for } j \text{ odd}, \end{cases}
$$

where $\bar{\gamma}_1$ and $\bar{\gamma}_2$ are arbitrary constants. This unfortunate fact is a direct consequence of the presence of the dispersive term in the second model equation (2.1a). However, if $r_0 > 0$, the dissipative term introduced in the first model equation (2.1a) enhances the stability of the scheme by controlling those unwanted oscillations. Indeed, if $r_0 > 0$, the last term of the energy is equal to zero if and only if the approximate strain is an affine function. Moreover, if $r_0 > 0$, the last term of the energy is infinity not only for the above approximate oscillating strain but also for approximate strains with highly oscillatory first derivatives

$$-\bar{\gamma}_{j-1} + \bar{\gamma}_j = \begin{cases} \bar{\gamma}_{x1} & \text{for } j \text{ even,} \\ \bar{\gamma}_{x2} & \text{for } j \text{ odd,} \end{cases}$$

and with highly oscillatory second derivatives

$$\bar{\gamma}_{j-1} - 2\bar{\gamma}_j + \bar{\gamma}_{j+1} = \begin{cases} \bar{\gamma}_{xx1} & \text{for } j \text{ even,} \\ \bar{\gamma}_{xx2} & \text{for } j \text{ odd.} \end{cases}$$

Since the energy must remain bounded, by Proposition 4.1, all three types of highly oscillatory waves in the approximate strain will be controlled, at least partially, even when $\omega = 0$. It is thus crucial to take $r_0 > 0$.

The effect of the size of ν_0. The stability result of Proposition 4.1 and the expression of the energy (4.4c) suggest that a good control on the variation of the strain and on the variation of its first and second derivatives can be achieved if we take $\beta = 1/2$ in the expression (2.5b) defining ν_0; namely, $\nu_0 = c_0\,(\Delta x)^{\beta}$. Taking β bigger than $1/2$ does not seem to be a reasonable strategy, because this would penalize too much the terms of the energy which control the oscillations of the strain. Taking β smaller than $1/2$ is certainly desirable, because then the computational cost, which is proportional to $(\Delta x)^{-3+\beta}$, decreases. However, in doing so, the control on the variation of the strain and on the variation of its first and second derivatives is relaxed, and mildly oscillatory waves might appear which can destroy the convergence to the exact solution.

5. Numerical experiments.

In this section we carry out several numerical experiments devised (i) to verify that the parameter r_0 must be taken to be strictly positive and (ii) to see for what values of the parameter β (recall we are taking $\nu_0 = c_0\,(\Delta x)^{\beta}$) the scheme converges. These experiments are done for the strain–stress relation of Fig. 1. We also report a convergence study for a piecewise-cubic strain–stress curve. Before presenting those numerical experiments, we perform a simple stability analysis that will guide us in the choice of the Courant–Friedrichs–Lewy (CFL) condition.

5.1. Linear stability analysis.

To have an idea of the stability condition for the above scheme, we study its linear stability. We thus consider linear stresses $\sigma(\gamma) = \sigma'\,\gamma$, and we assume that $\sigma' \geq 0$, since otherwise the resulting scheme is unconditionably unstable.

A simple computation gives the following form for the amplification matrix for the numerical scheme (3.1):

$$A = Id + \Lambda + \frac{1}{2}\Lambda^2 + \frac{1}{6}\Lambda^3,$$

where

$$\Lambda = \frac{\Delta t}{\Delta x}\begin{pmatrix} r_0\,\frac{\nu_0}{\Delta x}\,b & i\,a \\ i\sigma'\,a - i\,\frac{\lambda_0}{(\Delta x)^2}\,c & \frac{\nu_0}{\Delta x}\,b \end{pmatrix}$$

and

$$\begin{aligned} a &= \frac{2}{3}\sin(\theta/2)\cos(\theta/2)\,(3 + 2\sin^2(\theta/2)), \\ b &= -4\sin^2(\theta/2), \\ c &= -8\sin^3(\theta/2)\cos(\theta/2) \end{aligned}$$

for $\theta \in (-\pi, \pi]$. The eigenvalues of Λ are, taking into account (2.4),

$$\xi_\pm = \frac{\Delta t}{\Delta x}\left(\frac{\nu_0}{\Delta x}\right)\left\{\left(\frac{1+r_0}{2}\right)b \pm \left[\left(\frac{1+r_0}{2}\right)^2 b^2 - \Theta\right]^{1/2}\right\},$$

where

$$\Theta = r_0\,(b^2 + ac) + \left(\frac{\sigma'}{(\nu_0/\Delta x)^2}\right)a^2 - \left(\frac{1+r_0}{2}\right)^2 \omega^2\, ac.$$

Since $b \leq 0$, $b^2 + ac = \frac{16}{3}\big(\sin(\theta/2)\big)^6\big(1 + 2(\sin(\theta/2))^2\big) \geq 0$, and $ac \leq 0$, we have that $\Theta \geq 0$. Thus, the real part of the eigenvalues $\xi_\pm$ is always nonpositive, provided that $r_0 \geq 0$. Thus, the scheme is always linearly stable under a condition of the form

$$\frac{\Delta t}{\Delta x}\left(\frac{\nu_0}{\Delta x}\right) \leq \varphi\left(\omega, r_0, \frac{\sigma'}{(\nu_0/\Delta x)^2}\right).$$

We have numerically verified that the function

$$\varphi\left(\omega, r_0 = 1, \frac{\sigma'}{(\nu_0/\Delta x)^2}\right) \leq \phi(\omega) = \begin{cases} .44, & \omega \in [0, 1], \\ 1.848/(2.5 + \omega^{1.2}), & \omega \in (1, 50], \\ .8712/\omega, & \omega \in (50, 1000], \end{cases}$$

can be used when $\frac{\sigma'}{(\nu_0/\Delta x)} \leq 1$. In all our computations, we have chosen Δt as follows:

$$\Delta t = \Big\{ \sup_{\gamma \geq -1} \max\{0, \sigma'(\gamma)\}\Big\}^{-1/2} (\Delta x)^{2-\beta}\, \phi(\omega).$$

Note that $\Theta = 0$ if and only if $\theta = 0$ or $\theta = \pi$, and $r_0 = 0$. In this case, we have

$$\Lambda = \frac{\Delta t}{\Delta x}\begin{pmatrix} 0 & 0 \\ 0 & -4\frac{\nu_0}{\Delta x} \end{pmatrix},$$

and so, highly oscillatory waves of the form $\bar{\gamma}_j = \bar{\gamma}_0\,(-1)^j$ will never be damped away, regardless of the values of Δt, Δx, and ν_0. This fact agrees strongly with the analysis of the energy of the semidiscrete scheme.

5.2. Experiments for a piecewise linear stress–strain curve. Next, we present a study of the convergence properties of the scheme under consideration to the VC solution of (1.1) for the strain–stress curve and the initial data used at the introduction and the values $\omega = .15$, $\omega = 1$, and $\omega = 3$. Abeyaratne and Knowles [2] solved this Riemann problem. A classification of all the possible solutions of Riemann problems for the strain–stress relation of Fig. 1 (and general kinetic and nucleation criteria) was carried out by Abeyaratne and Knowles in [3]. The Riemann problem we have chosen is the most challenging among all the possible types of solutions since it has both phase boundaries and shocks.

The effect of the size of r_0. In Fig. 3, we compare the exact VC solution for $\omega = .15$ and the different approximate solutions given by the scheme, with $r_0 = 0$ and $\beta = 1/2$, for $\Delta x = 1/800, 1/1600, 1/3200, 1/6400$, and $1/12800$. In Fig. 4, we take $r_0 = 1$. We clearly see that the highly oscillatory waves that appear in the approximate strain for $r_0 = 0$ disappear

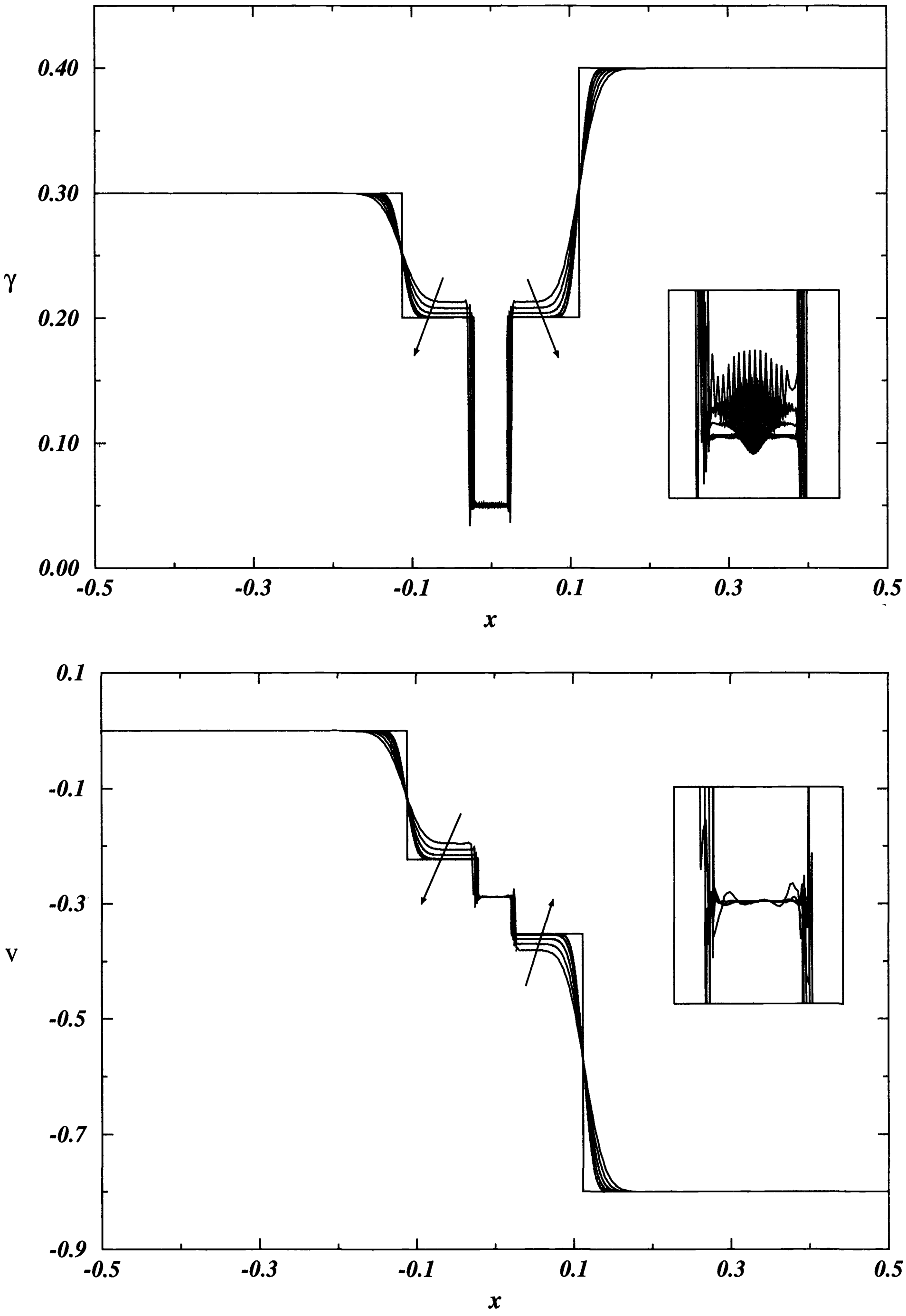


FIG. 3. *The choice $r_0 = 0$ (and $\beta = 1/2$) for $\omega = .15$.*

completely when $r_0 = 1$, as expected. The same phenomenon occurs for $\omega = 1$, as can be seen in Figs. 5 and 6, and for $\omega = 3$, as can be seen in Figs. 7 and 8. These results confirm that we must take $r_0 > 0$ and that choice (2.5a), $r_0 = 1$, is a reasonable one.

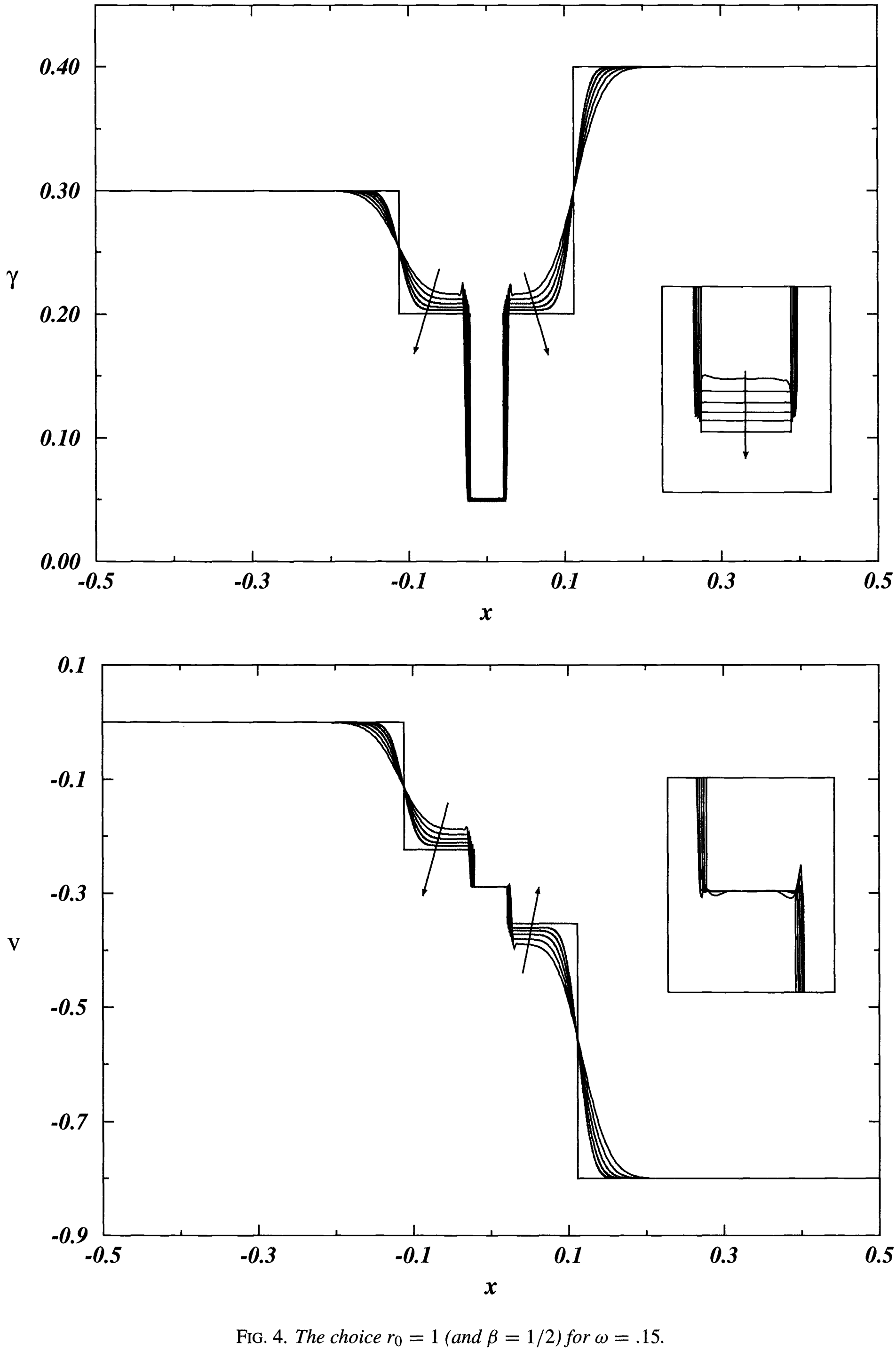


FIG. 4. *The choice $r_0 = 1$ (and $\beta = 1/2$) for $\omega = .15$.*

The effect of the size of β. In Figs. 9 and 10, we compare the exact VC solution for $\omega = .15$ and the different approximate solutions given by the scheme, with $r_0 = 1$ for $\Delta x = 1/800, 1/1600, 1/3200, 1/6400$, and $1/12800$. This is done for several values of β

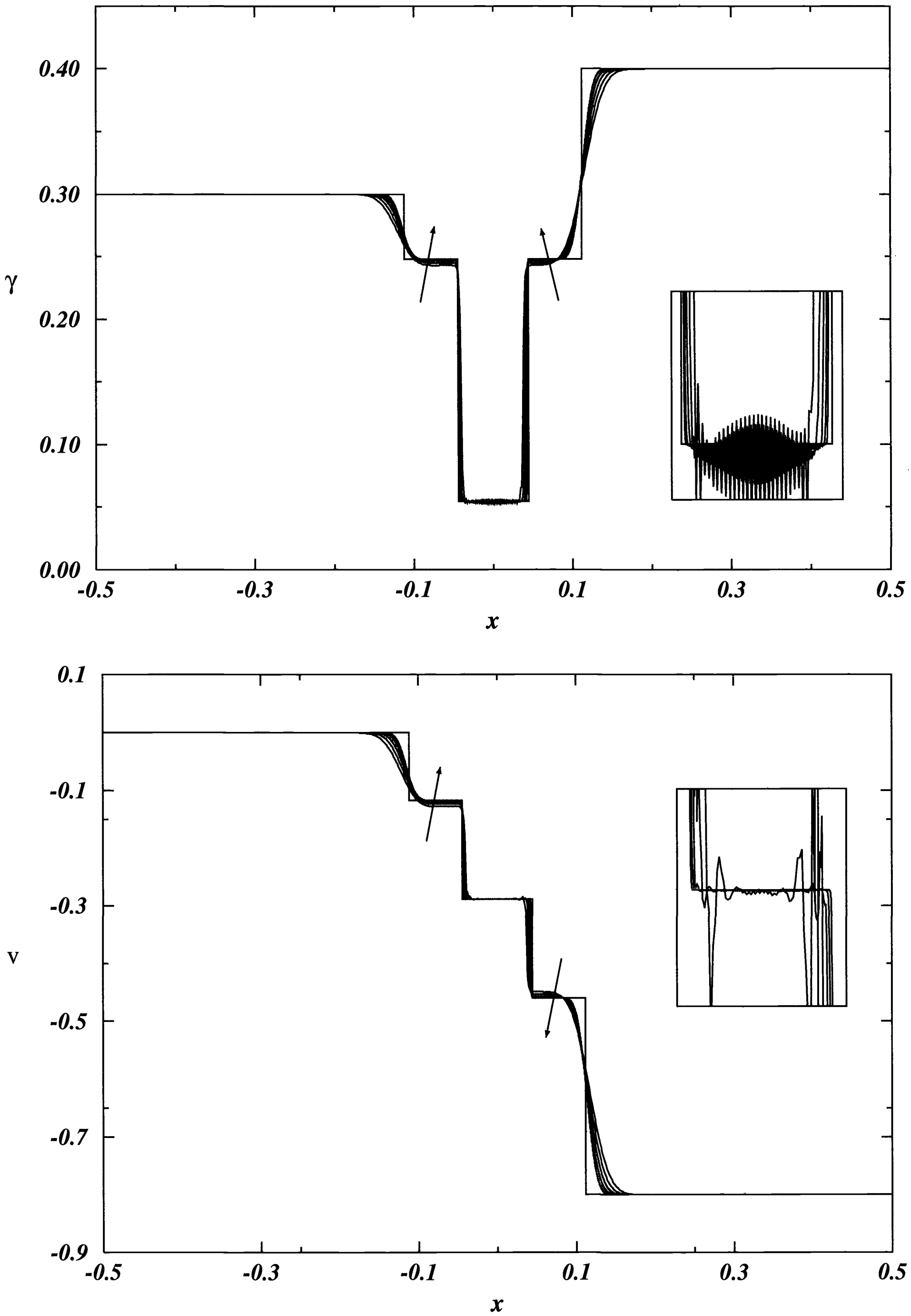


FIG. 5. *The choice* $r_0 = 0$ *(and* $\beta = 1/2$*) for* $\omega = 1$.

to see the effect of the parameter β on the convergence properties of the scheme. We clearly see that the scheme does not converge to the proper VC solution. We also see that as β decreases, the approximation properties of the scheme improve around the elliptic region.

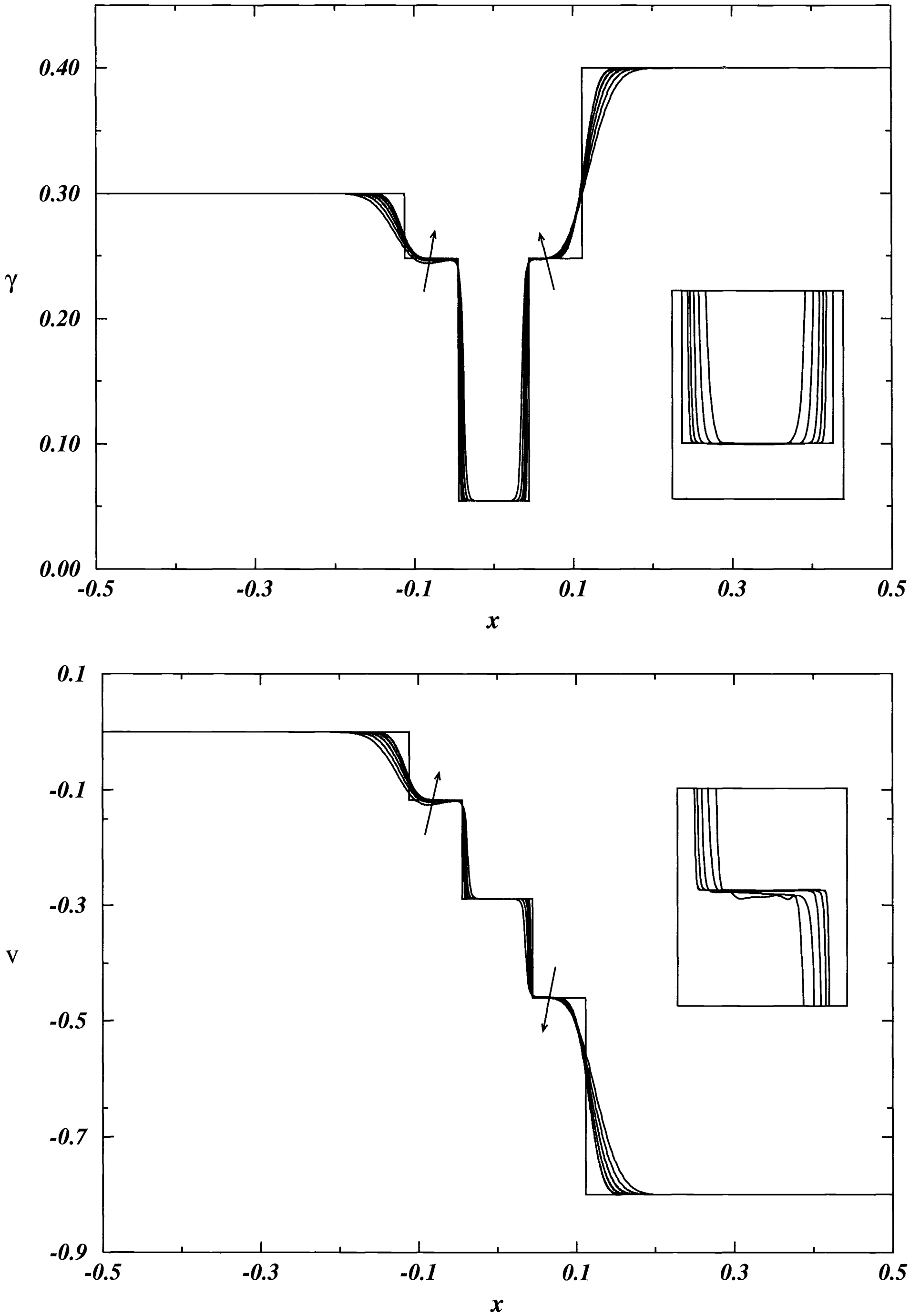


FIG. 6. *The choice* $r_0 = 1$ *(and* $\beta = 1/2$*) for* $\omega = 1$.

(Although they deteriorate around the hyperbolic region, as is expected, our main concern in this paper is to assess the quality of the approximation around the elliptic region, since it is very well known how to improve the quality of the approximation on the hyperbolic region.)

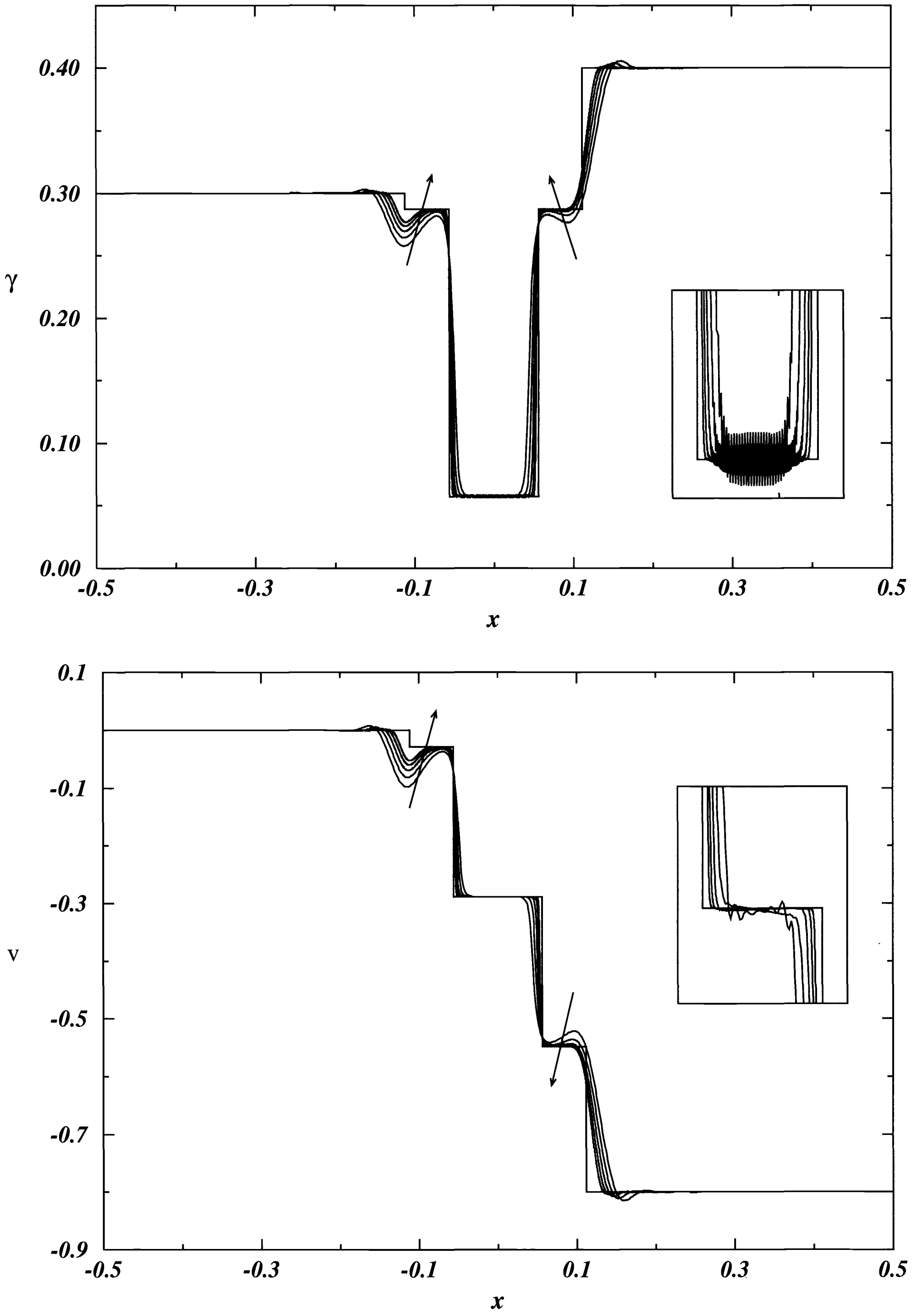


FIG. 7. *The choice* $r_0 = 0$ *(and* $\beta = 1/2$*) for* $\omega = 3$.

We have verified that the same phenomenon takes place for $\omega = 1$ and $\omega = 3$, but we are only displaying the results for $\omega = .15$ because for small ω this phenomenon is accentuated. A

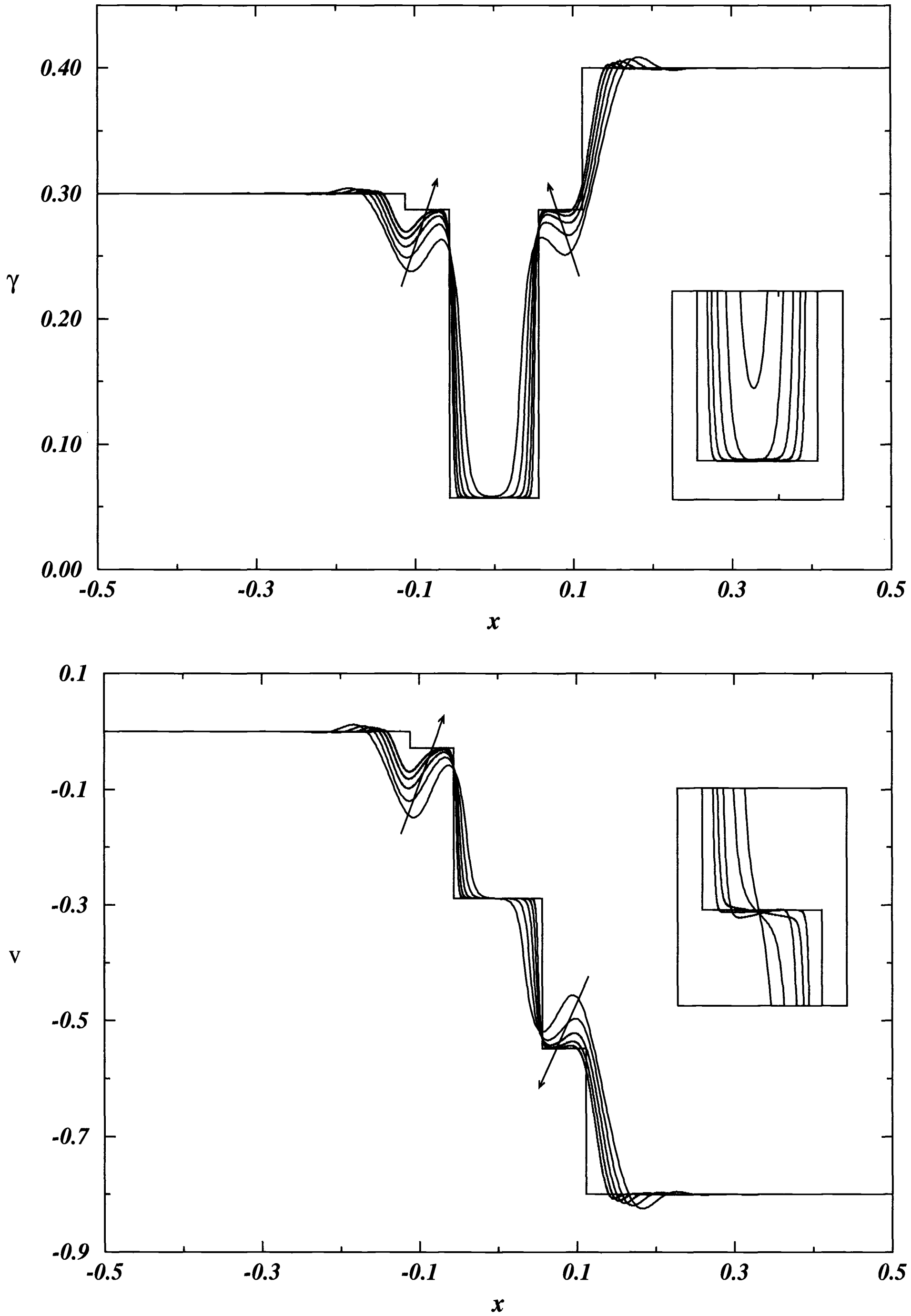


FIG. 8. *The choice* $r_0 = 1$ *(and* $\beta = 1/2$*) for* $\omega = 3$.

more precise assessment of this phenomenon is shown in Figs. 11, 12, and 13, where the case $\Delta x = 1/25600$ has been included.

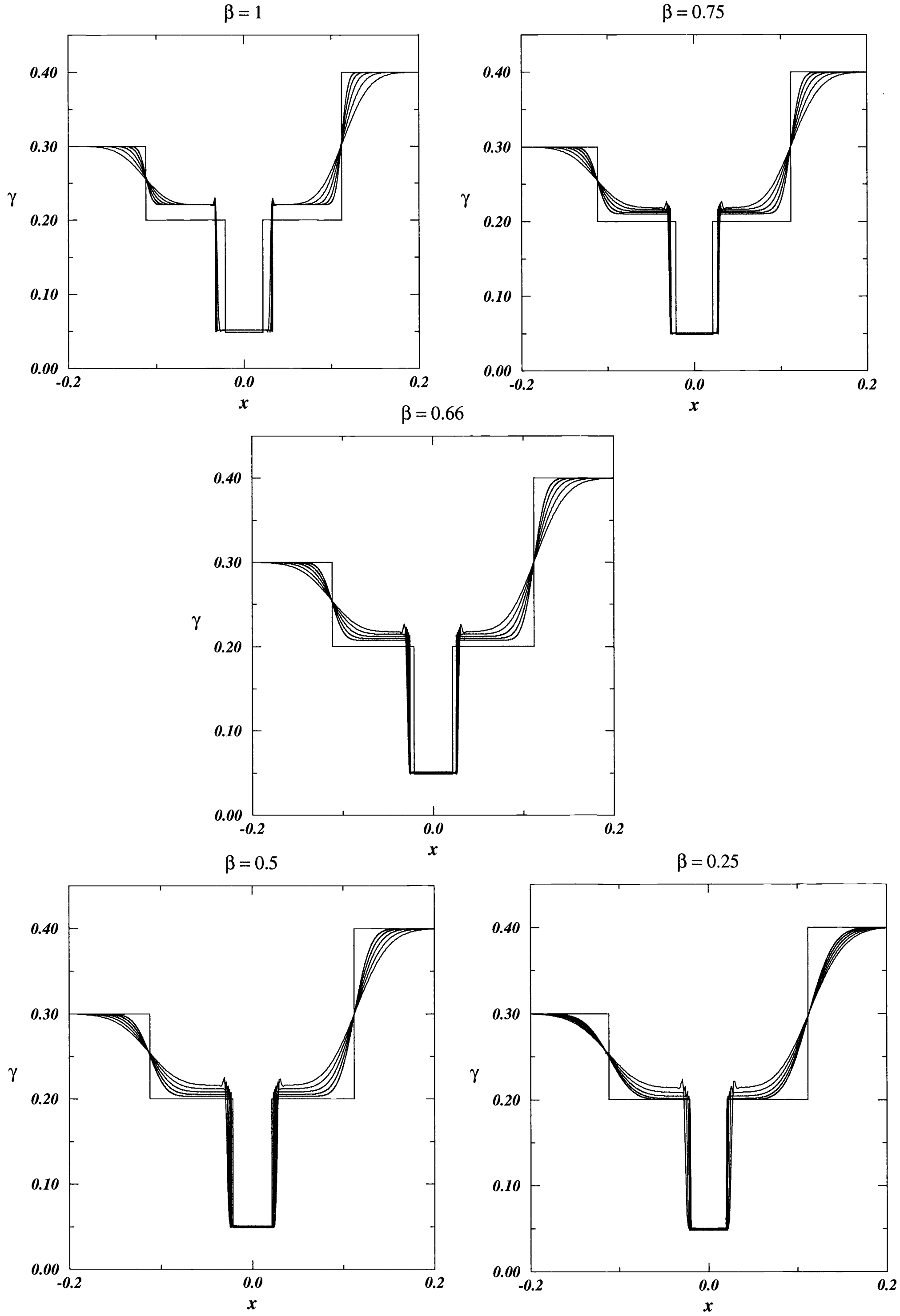


FIG. 9. *The effect of β: exact and approximate strains for $\omega = .15$.*

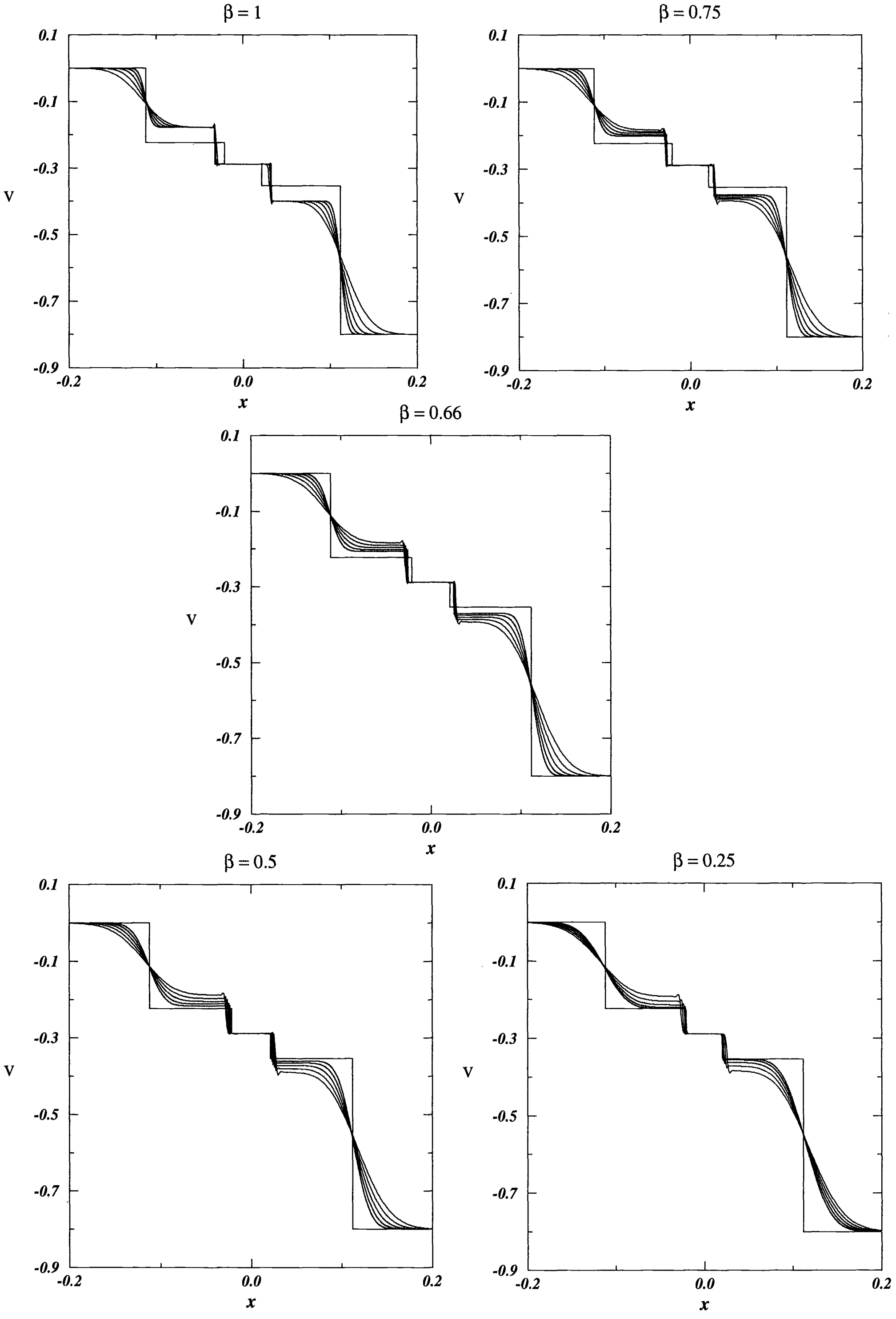


FIG. 10. *The effect of β: exact and approximate velocities for $\omega = .15$.*

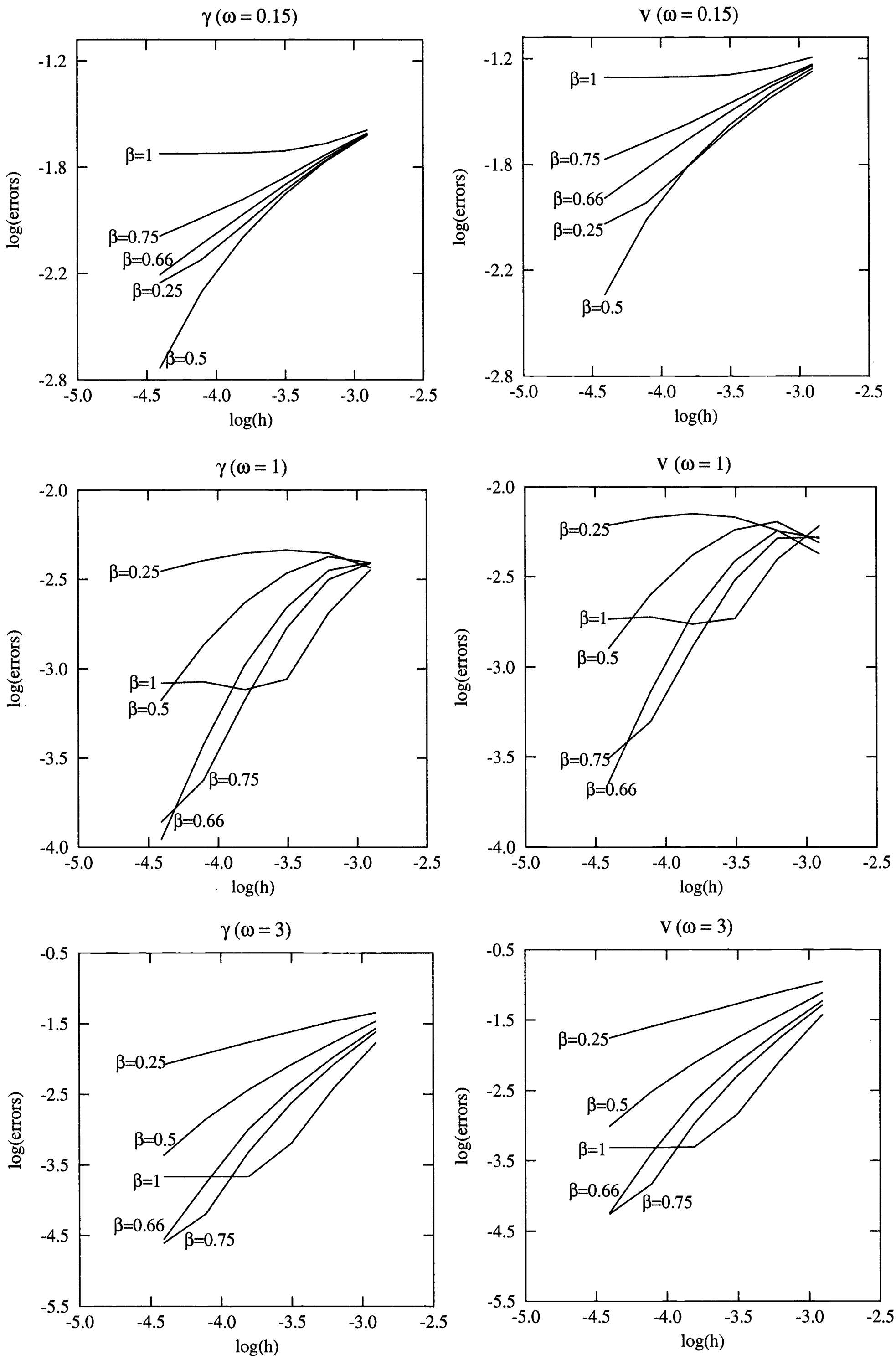


FIG. 11. *History of convergence of the average of the error on the region $R_{r.\ ph.\ b.}$ to the right of a phase boundary.*

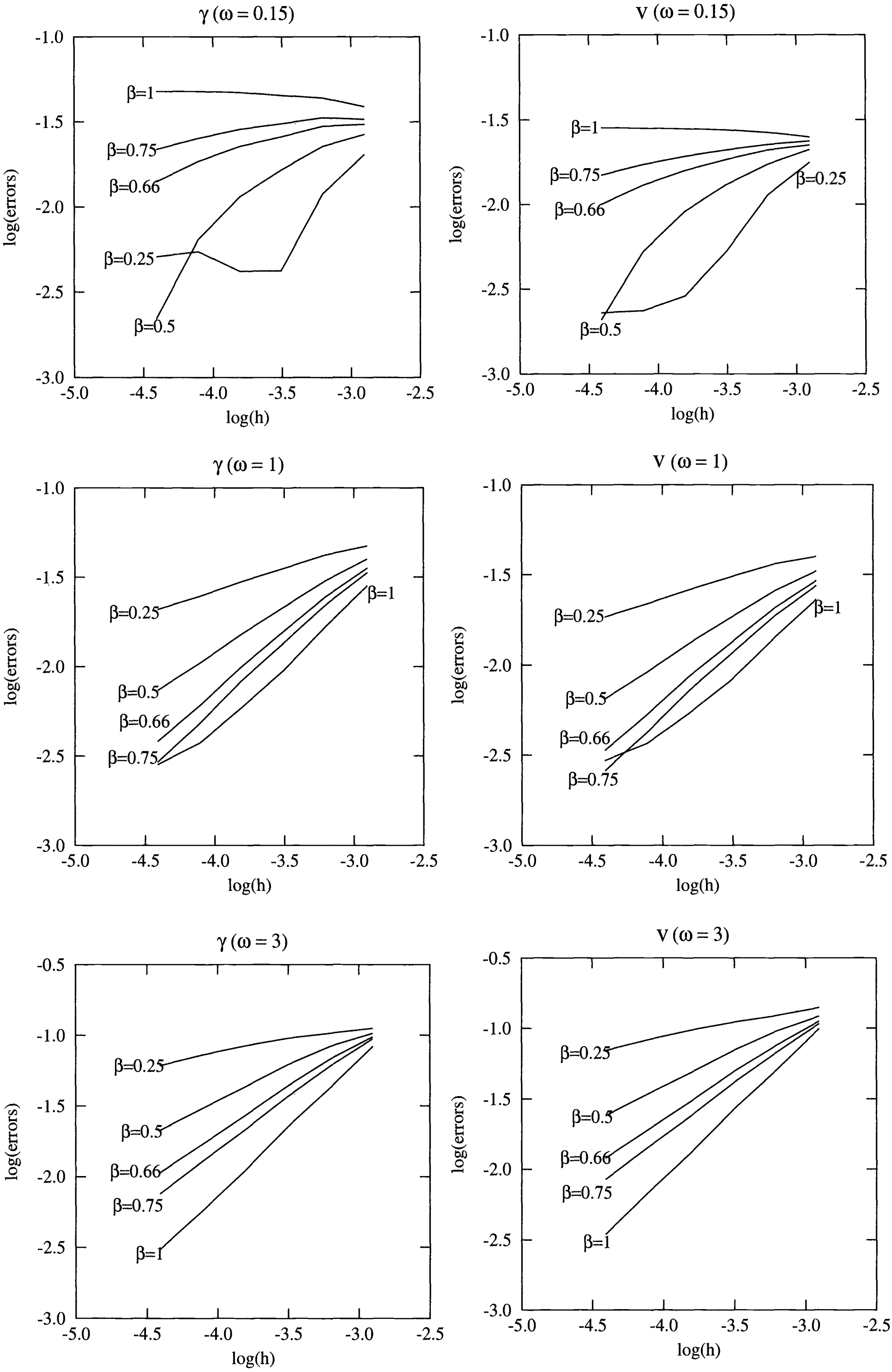


FIG. 12. *History of convergence of the average of the error on the region $R_{ph.\,b.}$ around a phase boundary.*

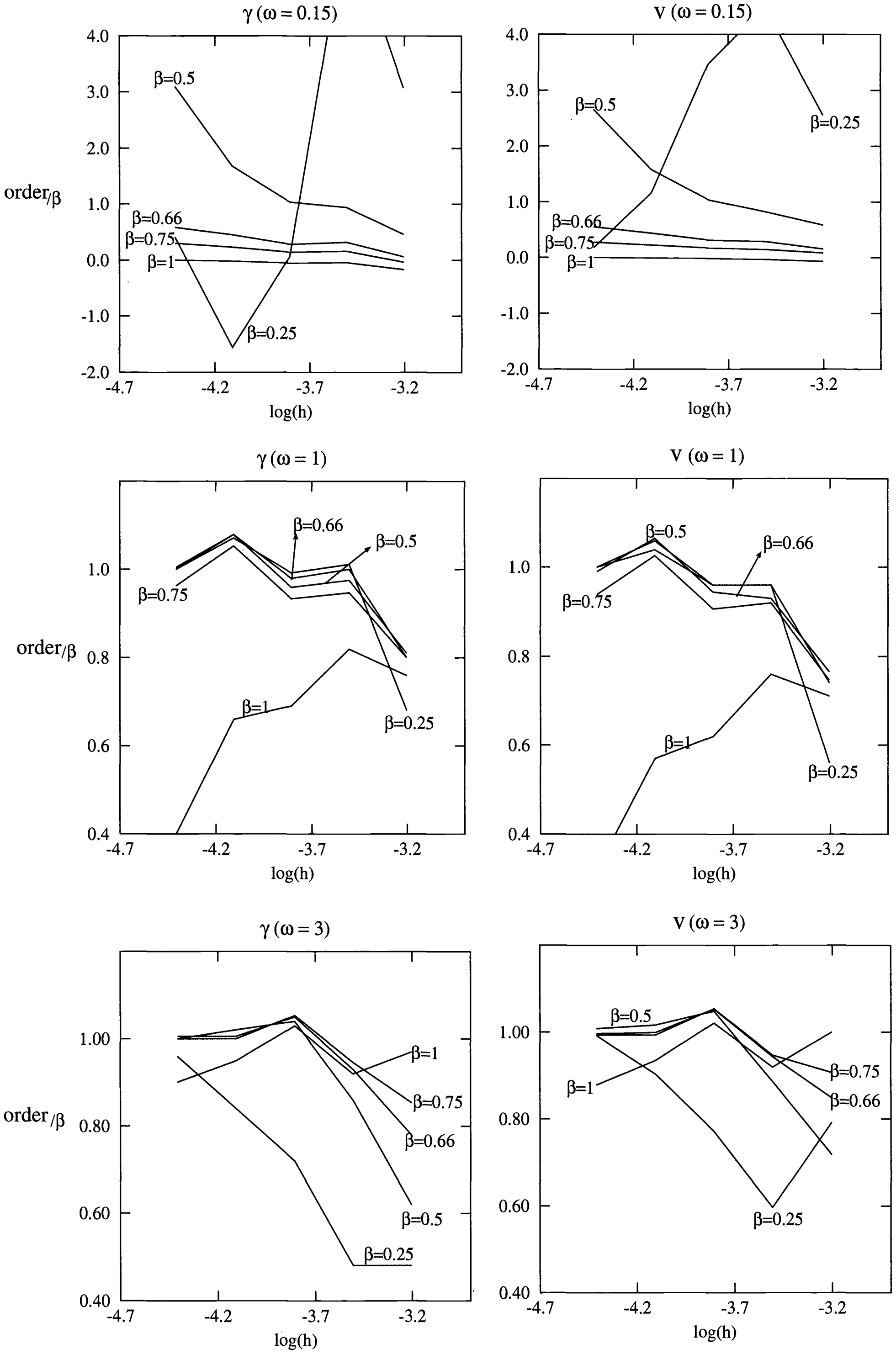


FIG. 13. *History of the ratio (numerical order of convergence)/β.*

In Fig. 11, we show the history of convergence of the scheme on the region $R_{r.\ ph.\ b.}$ which is a region located at the right of the rightmost phase boundary. The region $R_{r.\ ph.\ b.}$ is defined as follows:

$$R_{r.\ ph.\ b.} = \begin{cases} (.041, .092), & \omega = .15, \\ (.064, .092), & \omega = 1, \\ (.076, .092), & \omega = 3. \end{cases}$$

The average of the error on the region $R_{r.\ ph.\ b.}$ is plotted as a function of Δx for different values of β and ω. Again, we see that the scheme converges well for $\beta = 1/2$ but does not converge for $\beta = 1$. Notice how this feature becomes more acute as ω decreases. Also note that the scheme begins to be noticeably nonconvergent when the error is of the order of 10^{-2} for $\omega = .15$, 10^{-3} for $\omega = 1$, and 10^{-4} for $\omega = 3$. Thus, this phenomenon may not be appreciated on standard pictures.

In Fig. 12, we show the history of convergence of the scheme on the region $R_{ph.\ b.}$ which is a small region around one of the rightmost phase boundaries. The region $R_{ph.\ b.}$ is defined as follows:

$$R_{ph.\ b.} = \begin{cases} (.001, .041), & \omega = .15, \\ (.024, .064), & \omega = 1, \\ (.036, .076), & \omega = 3. \end{cases}$$

Theoretically, the width of the boundary layer around a phase boundary of the solution of (1.2) is of the order of the viscosity coefficient ν; see [9], [2], and [4]. Since we have constructed our approximate solution as a good approximation of the solutions (1.2), the width of the boundary layer around a phase boundary of our approximate solutions should be of order $(1 + r_0)\,\nu_0$, which by assumption (2.5b) is of order $(\Delta x)^{\beta}$ as Δx goes to zero. Thus, if the scheme is converging well around the phase boundary, the estimated rate of convergence of the average of the error on $R_{ph.\ b.}$ should be very similar to the parameter β. We display the ratios of these numbers in Fig. 13.

We can see very good agreement with the theoretical order of convergence for $\beta = .5$ and $\omega = 1$ and $\omega = 3$; for $\beta = .5$ and $\omega = .15$, the width of the boundary layer around the phase boundary is much smaller than expected. (We conjecture that this is so because one of the values of the exact strain at the phase boundary is .20017..., which is extremely close to the value $\gamma_{2,3} = .2$ that separates phases 2 and 3.) For $\beta = 1$ and $\omega = 3$, we see that for Δx the ratio of the orders of convergence decreases as Δx decreases. This unfortunate situation is accentuated as the value of ω decreases. As β varies from 1/2 to 1, the behavior of the ratios of orders of convergence seems to vary almost monotonically with β. In the case $\beta = .25$ the asymptotic regime does not seem to have been achieved most probably because there is an enormous amount of artificial viscosity.

5.3. A piecewise-cubic stress–strain curve. Finally, we present a short study of the convergence properties of the scheme under consideration to the VC solution of (1.1) for the strain–stress curve

$$\sigma(\gamma) = \gamma\,(\gamma - .5)\,(\gamma - 1);$$

see Fig. 14, $\omega = .15$, $\omega = 1$, and $\omega = 3$. The initial data is the following:

$$\gamma_0(x) = \begin{cases} 1.072649343, & x < 0 \\ .15, & x > 0 \end{cases}, \qquad v_0(x) \equiv 0.$$

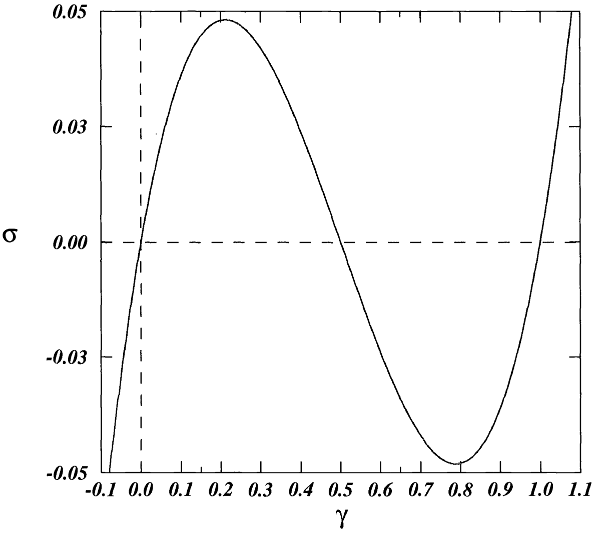


FIG. 14. *The piecewise-cubic strain–stress relation.*

The results are very similar to those of the piecewise linear stress–strain curve; see Figs. 15–17. Moreover, as Δx goes to zero, the numerical order of convergence gets closer and closer to the value $\beta = .5$ for all the values of ω. We do not observe the unusually thin boundary layer around the phase transition for $\omega = .15$ observed in the case of the piecewise-linear stress–strain curve. (This strengthens our conviction that the thinning of the numerical boundary layer is due to the closeness of one of the values of the strain to the elliptic region.)

6. Conclusion. In this paper, we continue the work initiated by Slemrod and Flaherty [11] and devise and numerically test a simple finite-difference method for approximating the VC solutions of the Cauchy problem for the mixed-type system governing the propagation of phase transitions in solids (or in van der Waals fluids). The numerical experiments show that (i) the scheme produces approximate velocities that do not display the spike at the phase transitions reported in [11] and (ii) it converges for all positive values of the VC parameter ω when the artificial viscosity is taken to be proportional to $(\Delta x)^{1/2}$ but fails to converge if it is taken proportional to Δx. This shows that an incorrect discretization procedure can significantly alter the nature of the solutions to which the scheme is converging.

The next step in the development of numerical methods for (1.1) is to find out how to use the heuristics introduced in this paper to devise formally high-order accurate (finite difference, finite volume, or relaxation) methods that incorporate the various techniques (characteristicwise evaluation of the fluxes, slope limiting, etc.) that work well for purely hyperbolic systems.

Acknowledgments. The authors thank Rohan Abeyaratne, who sparked the interest of the authors in this topic, Richard James for fruitful discussions about the spikes on the approximate velocity, Lev Truskinovsky for stimulating input and for providing the exact solution of the numerical experiment in §5, and Shi Jin for his incisive and constructive remarks. They also thank John Lowengrub for fruitful conversations.

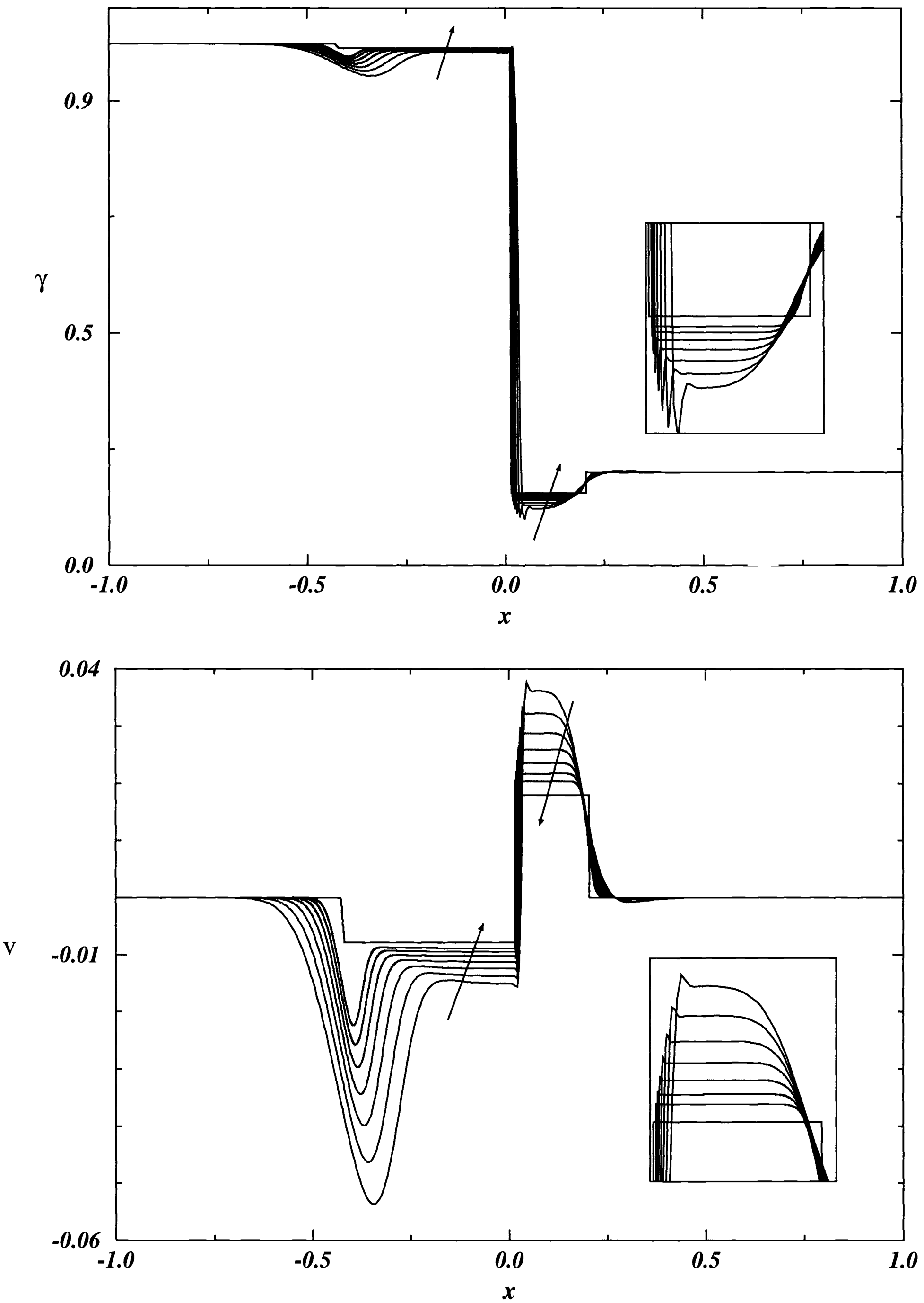


FIG. 15. *Exact and approximate solutions for* $\omega = .15$, $\beta = 1/2$.

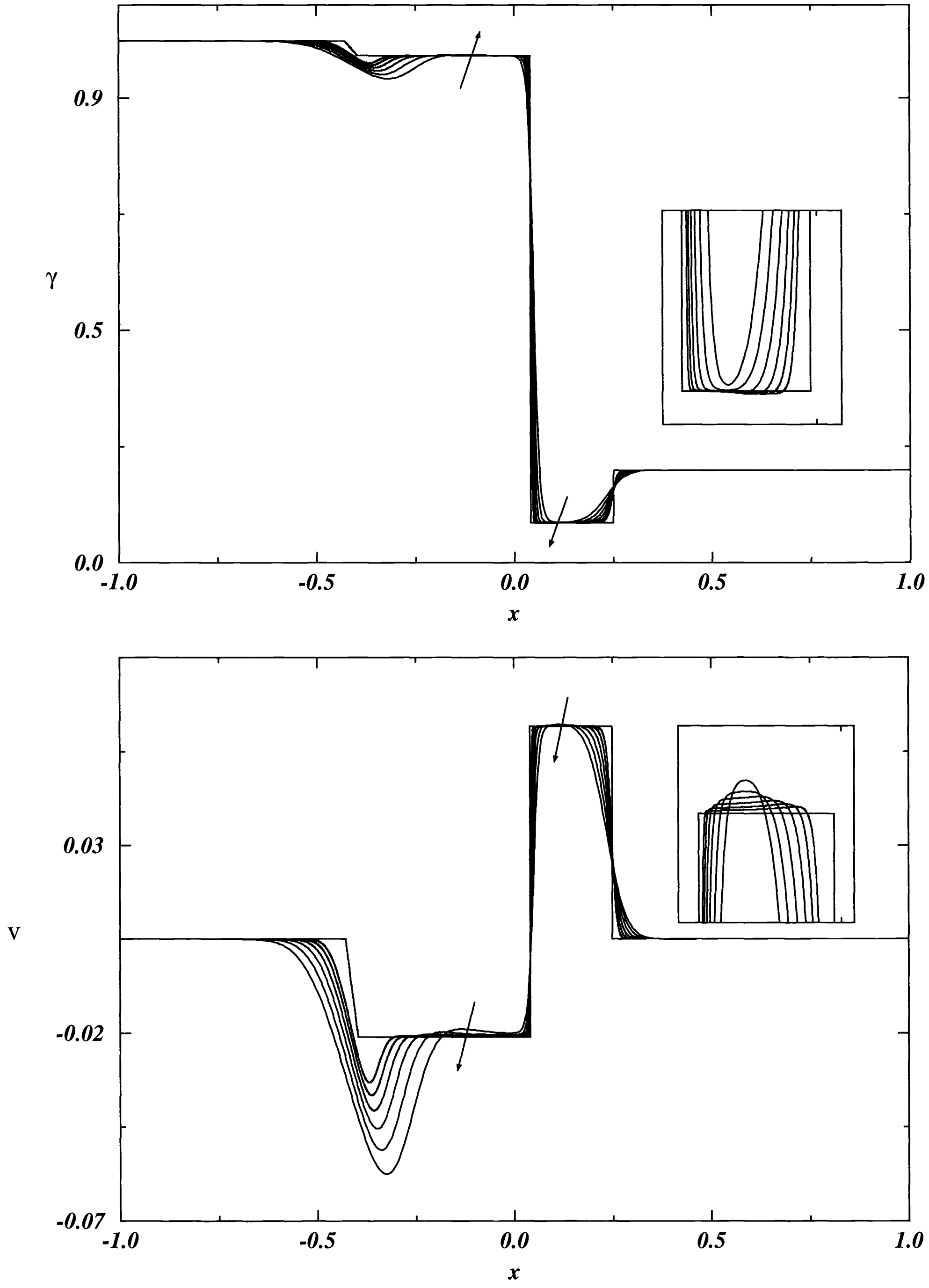
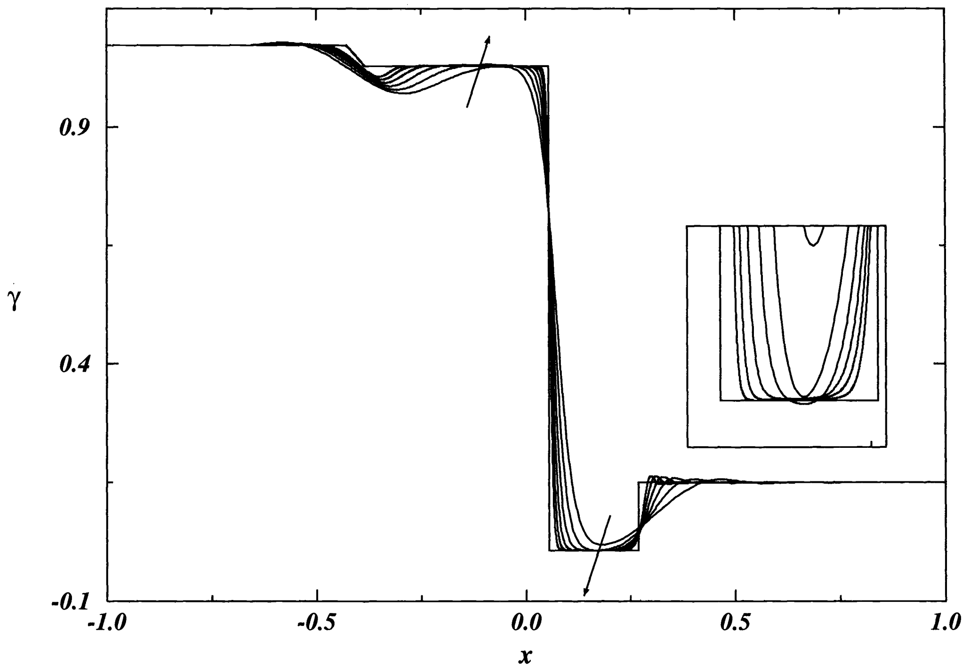


FIG. 16. *Exact and approximate solutions for* $\omega = 1$, $\beta = 1/2$.



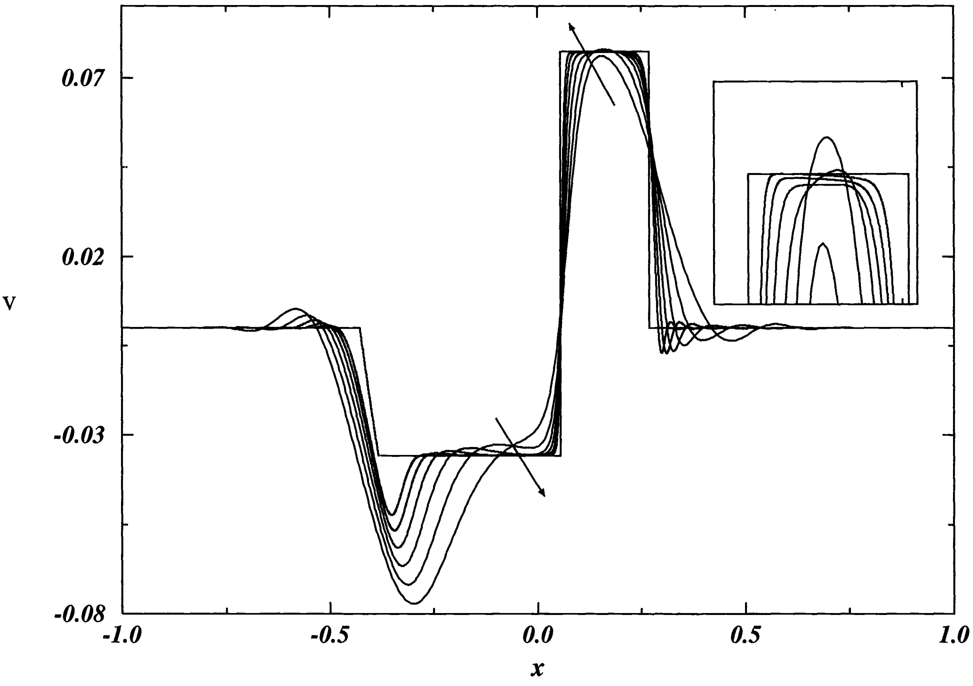


FIG. 17. *Exact and approximate solutions for $\omega = 3$, $\beta = 1/2$.*

REFERENCES

[1] R. Abeyaratne and J. K. Knowles, *Shock induced transitions and phase structures in general media*, in The IMA Volumes in Mathematics and its Applications, J. E. Dunn, R. Fosdick, and M. Slemrod, eds., Springer-Verlag, Berlin, New York, 1993, pp. 185–229.

[2] ———, *Implications of viscosity and strain gradient effects for the kinetics of propagating phase boundaries in solids*, SIAM J. Appl. Math., 51 (1991), pp. 1205–1221.

[3] ———, *Kinetic relations and the propagation of phase boundaries in solids*, Arch. Rational Mech. Anal., 114 (1991), pp. 119–154.

[4] M. Affouf and R. Caflish, *A numerical study of Riemann problem solutions and stability for a system of viscous conservation laws of mixed type*, SIAM J. Appl. Math., 4 (1991), pp. 605–634.

[5] H. Fan and M. Slemrod, *Shock induced transitions and phase structures in general media*, in The IMA Volumes in Mathematics and its Applications, J. E. Dunn, R. Fosdick, and M. Slemrod, eds., Springer-Verlag, Berlin, New York, 1993, pp. 61–91.

[6] R. James, *The propagation of phase boundaries in elastic bars*, Arch. Rational Mech. Anal., 73 (1980), pp. 125–158.

[7] S. Jin, *Numerical integrations of systems of conservation laws of mixed type*, SIAM J. Appl. Math., 55 (1995), pp. 1536–1551.

[8] E. B. Pitman and Y. Ni, *Visco-elastic relaxation with a van der Waals type stress*, Internat. J. Engrg. Sci., 32 (1994), pp. 327–338.

[9] M. Slemrod, *Physical Partial Differential Equations*, J. Lightbourne and S. Rankin, eds., Marcel Dekker, New York, 1984, pp. 75–84.

[10] ———, *Dynamic phase transitions in a van der Waals fluid*, J. Differential Equations, 52 (1984), pp. 1–23.

[11] M. Slemrod and J. E. Flaherty, *Phase transformations*, E. C. Aifantis and J. Gittus, eds., Elsevier Applied Science Publishers, New York, 1986, pp. 203–212.

[12] C.-W. Shu, *A numerical method for systems of conservation laws of mixed type admitting hyperbolic flux splitting*, J. Comput. Phys., 100 (1992), pp. 424–429.

[13] C.-W. Shu and S. Osher, *Efficient implementation of essentially non-oscillatory shock-capturing methods*, J. Comput. Phys., 77 (1988), pp. 439–471.

[14] L. Truskinovsky, *Equilibrium phase interfaces*, Dokl. Akad. Nauk. uZSSSR, 256 (1982), pp. 306–310.

[15] ———, *Dynamics of non-equilibrium phase boundaries in a heat conducting non-linearly elastic medium*, PMM U.S.S.R., 51 (1987), pp. 777–784.

[16] ———, *Shock induced transitions and phase structures in general media*, in The IMA Volumes in Mathematics and its Applications, J. E. Dunn, R. Fosdick, and M. Slemrod, eds., Springer-Verlag, Berlin, New York, 1993, pp. 185–229.

SIAM J. SCI. COMPUT.
Vol. 17, No. 5, pp. 1122–1134, September 1996

A TWO-DIMENSIONAL COMPOSITE GRID NUMERICAL MODEL BASED ON THE REDUCED SYSTEM FOR OCEANOGRAPHY*

Y. F. XIE[†], G. L. BROWNING[†], AND G. CHESSHIRE[‡]

Abstract. The proper mathematical limit of a hyperbolic system with multiple time scales, the reduced system, is a system that contains no high-frequency motions and is well posed if suitable boundary conditions are chosen for the initial-boundary value problem. The composite grid method, a robust and efficient grid-generation technique that smoothly and accurately treats general irregular boundaries, is used to approximate the two-dimensional version of the reduced system for oceanography on irregular ocean basins. A change-of-variable technique that substantially increases the accuracy of the model and a method for efficiently solving the elliptic equation for the geopotential are discussed. Numerical results are presented for circular and kidney-shaped basins by using a set of analytic solutions constructed in this paper.

Key words. composite mesh, shallow-water equations, reduced systems, stability, ocean basins

AMS subject classifications. 65C20, 65M50, 76D05

1. Introduction.

Kreiss [11] introduced the concept of a smooth solution of a hyperbolic system with multiple time scales. The smooth motions are of interest to meteorologists and oceanographers because they carry the majority of the energy. In the original theory, the existence of a well-posed limit system, the reduced system, which could be used to accurately describe the smooth solutions, was also proved. The limit procedure removes the high-frequency motions from the original hyperbolic system. A reduced system for three-dimensional oceanic flows has recently been introduced (Browning, Holland, and Kreiss [2]) and numerical solutions of this system in a channel for a single scale of motion, i.e., a mesoscale eddy, have been obtained (Browning et al. [3]). In the latter study a uniform mesh was appropriate; however, in a full ocean basin where the spatial scales vary widely, a uniform mesh is inappropriate. The composite grid method described in detail by Chesshire and Henshaw [9] treats an irregular boundary smoothly and accurately, places fine resolution only where it is needed, and is flexible for a multiple mesh configuration because interpolation is used on the overlapping regions. Therefore, the composite grid method is an ideal candidate for solving the multiscale oceanic problem in an accurate and efficient manner.

To implement the reduced system by using the composite grid method for ocean circulation studies, we introduce a change-of-variable technique to overcome the truncation error imbalance in the system and a method to efficiently and accurately solve the elliptic equation of the system. The efficiency and accuracy of the reduced system when using the composite grid method are demonstrated by constructing a set of analytic solutions of the shallow-water equations with different scales and boundary configurations. In §2 the reduced system for the shallow-water equations is introduced, and the absence of high-frequency motions is proved. We construct analytic solutions of the full shallow-water equations on circular and kidney-shaped domains in §3. On the basis of a truncation error analysis, a change-of-variable technique is introduced. In §4, we generate overlapping grids for circular and kidney-shaped domains by using the composite grid method. An efficient method for the direct solution of the elliptic equation with Neumann boundary conditions for the geopotential is proposed for

*Received by the editors December 5, 1994; accepted for publication (in revised form) May 25, 1995.

[†]CIRA, Colorado State University, Foothills Campus, Fort Collins, CO 80523 (xiey@fsl.noaa.gov, browning@fsl.noaa.gov). These authors were supported by the Office of Naval Research under contract ONR-N00014-92-J-1089 and NOAA/ERL Forecast Systems Laboratory, 325 Broadway, Boulder, CO 80303.

[‡]Los Alamos National Laboratory, Los Alamos, NM 87545 (geoff@lanl.gov). The author was supported by the Office of Naval Research under contract ONR-N00014-91-C-0081.

the reduced system. In §5, we present numerical results and discuss a numerical technique to control the numerical divergence. Finally, our conclusions are presented in §6.

2. The reduced system. The shallow-water equations are frequently used to describe atmospheric and oceanic motions. In Cartesian coordinates x and y, directed eastward and northward, respectively, the shallow-water equations are (e.g., [4])

$$\frac{du}{dt} + \phi_x - fv = 0, \tag{2.1a}$$

$$\frac{dv}{dt} + \phi_y + fu = 0, \tag{2.1b}$$

$$\frac{d\phi}{dt} + (\phi_0 + \phi)(u_x + v_y) = 0, \tag{2.1c}$$

where t is time, u and v are the velocity components in the x- and y-directions, ϕ_0 is the mean geopotential, ϕ is the deviation of the geopotential from the mean, and f is the Coriolis parameter that is assumed to have the constant value $f = 10^{-4}$. The total differential operator d/dt is defined as

$$\frac{d}{dt} = \frac{\partial}{\partial t} + u\frac{\partial}{\partial x} + v\frac{\partial}{\partial y}.$$

The shallow-water equations contain two classes of motions, i.e., the low-frequency (or Rossby) motions and the high-frequency (or inertial-gravity) motions (see, e.g., Browning, Kasahara, and Kreiss [4]). The majority of the energy of atmospheric and oceanographic flows is contained in the low-frequency motions but the high-frequency motions in the system adversely affect the Courant–Freidrichs–Lewy (CFL) stability criterion and physical parameterizations. Therefore it is desirable to use a system where the high-frequency motions are not present.

Kreiss [11] showed that the low-frequency motions (the smooth solutions) can be approximated by using the proper mathematical limit called the reduced system, which is automatically well posed. The proper limit of the full system (2.1) in the external mode case is

$$\frac{du}{dt} + \phi_x - fv = 0, \tag{2.2a}$$

$$\frac{dv}{dt} + \phi_y + fu = 0, \tag{2.2b}$$

$$u_x + v_y = 0, \tag{2.2c}$$

which to first approximation accurately determines the smooth solution of the original system. To simplify the discussion, we define the Jacobian for any functions (x', y') of (x, y) as

$$J(x, y; x', y') = \begin{pmatrix} \frac{\partial x'}{\partial x} & \frac{\partial x'}{\partial y} \\ \frac{\partial y'}{\partial x} & \frac{\partial y'}{\partial y} \end{pmatrix}.$$

If (2.2a) and (2.2b) are appropriately differentiated, an elliptic equation can be derived from (2.2c) to determine ϕ:

$$\nabla^2\phi = f\zeta + 2\ \det[J(x, y; u, v)], \tag{2.3}$$

where $\zeta = -u_y + v_x$ is the vertical component of the vorticity.

In addition to being well posed, the reduced system has another important feature, namely, no high-frequency waves. To see this, assume the solution of the reduced system (2.2) to be

2π-periodic in the x- and y-directions and consider the linear constant coefficient system

$$u_t + \bar{u}u_x + \bar{v}u_y + \phi_x - fv = 0, \tag{2.4a}$$

$$v_t + \bar{u}v_x + \bar{v}v_y + \phi_y + fu = 0, \tag{2.4b}$$

$$u_x + v_y = 0, \tag{2.4c}$$

where $\bar{u}$ and $\bar{v}$ are given constants. Cross differentiating (2.4a) and (2.4b), one determines ϕ by

$$\nabla^2\phi - f\zeta = 0; \tag{2.5}$$

i.e., for this linear system, the Jacobian term disappears in the elliptic equation. Fourier transforming in space yields the system of ordinary differential equations

$$\hat{u}_t + i\bar{u}k\hat{u} + i\bar{v}l\hat{u} + ik\hat{\phi} - f\hat{v} = 0,$$

$$\hat{v}_t + i\bar{u}k\hat{v} + i\bar{v}l\hat{v} + il\hat{\phi} + f\hat{u} = 0,$$

$$-(k^2 + l^2)\hat{\phi} - f(-il\hat{u} + ik\hat{v}) = 0,$$

where k and l are the wave numbers corresponding to x and y, respectively. If the last equation is solved for $\hat{\phi}$ and substituted into the first two equations, the system can be written in the vector form

$$\hat{V}_t = \left[i(\bar{u}k + \bar{v}l)I + \frac{f}{k^2 + l^2}A \right] \hat{V},$$

where $\hat{V} = (\hat{u}, \hat{v})^T$, I is the identity matrix, and

$$A = \begin{pmatrix} -kl & -l^2 \\ k^2 & kl \end{pmatrix}.$$

Because both of the eigenvalues of A are zero, both of the eigenvalues of the coefficient matrix of $\hat{V}$ are $i(\bar{u}k + \bar{v}l)$; i.e., only the slow time scale is contained in this system (compare with the analysis of the full shallow-water equations in [4]).

For the reduced system, an elliptic solver is needed to determine ϕ, but a much larger time step can be used because the high-frequency waves are absent. For example, if a second-order in space and in time finite difference scheme is used, the stability of the linearized reduced and full systems requires the time steps to be

$$\Delta t \le \frac{\Delta x}{2 + f\Delta x},$$

$$\Delta t \le \frac{\Delta x}{2 + \sqrt{\Delta x^2 f^2 + 2\phi_0}},$$

respectively (assuming $\Delta y = \Delta x$). Because $\phi_0 = gh_0$, where $g \simeq 10ms^{-1}$ is the gravity acceleration and h_0 is the mean depth, $\phi_0 \simeq 10^3$ if $h_0 = 100$ m or $\phi_0 \simeq 10^4$ if $h_0 = 1$ km for the ocean problem. Thus the time step for the full system must be much smaller than the one for the reduced system. For example, when $\Delta x = 10^4$ m and $\phi_0 = 10^4$, the time step for the full system must be 100 times smaller. The longer time step for the reduced system can compensate for the additional expense of the solution of the elliptic equation for the geopotential if an elliptic equation solver is efficient (the elliptic solver takes only about two times longer than the hyperbolic part for the test problems presented in §5). Therefore, the reduced system is chosen for solving the oceanographic problem because it can be solved very efficiently compared to the full system.

3. Analytic solutions and change-of-variable technique. To construct an analytic solution on a relatively realistic basin, an analytic solution of the shallow-water equations on a kidney-shaped region (the East Coast of the United States looks like the west boundary of the kidney-shaped basin) is constructed to demonstrate the efficiency and accuracy of the reduced system when using the composite grid method. An analytic steady-state solution on a circular basin is mapped to a kidney-shaped region. This invertible mapping is used to map the solution (u, v, ϕ) and the shallow-water equations on the circle onto the kidney-shaped region; i.e., the transformed variables are an analytic solution of the transformed shallow-water equations. Although the transformed streamfunction Ψ looks realistic, the transformed solution is not a solution of the untransformed shallow-water equations on the kidney-shaped region. By mapping only the streamfunction Ψ of the circular region onto the kidney-shaped region, one can form a solution of the untransformed forced shallow-water equations on the kidney-shaped domain. A similar technique of constructing an analytic solution of a forced system is used in [1].

By using polar coordinates (i.e., $x = r\cos\theta$ and $y = r\sin\theta$), one can obtain a set of analytic solutions of the full shallow-water equations (1) on the circular domain. Let us choose one particular solution from this set of solutions for the circular basin with radius $R = 2500$ km:

$$u = -\left[\left(\frac{x}{R}\right)^2 + \left(\frac{y}{R}\right)^2\right]\frac{y}{R}, \tag{3.6a}$$

$$v = \left[\left(\frac{x}{R}\right)^2 + \left(\frac{y}{R}\right)^2\right]\frac{x}{R}, \tag{3.6b}$$

$$\phi = \frac{1}{6}\left[\left(\frac{x}{R}\right)^2 + \left(\frac{y}{R}\right)^2\right]^3 + \frac{fR}{4}\left[\left(\frac{x}{R}\right)^2 + \left(\frac{y}{R}\right)^2\right]^2. \tag{3.6c}$$

Using the solutions, we can derive an analytic solution for the shallow-water equations in a more realistic ocean basin by constructing an invertible mapping between a circle and a kidney-shaped region. A modified cardioid mapping

$$x' = (0.8 + 0.5x)^2 - (0.75y)^2,$$
$$y' = 2(0.8 + 0.5x)(0.75y)$$

is used to map any function from a circular region to a kidney-shaped region shown in Fig. 1. The Jacobian of this mapping is bounded away from zero $(\det[J(x, y; x', y')] \geq 0.135)$ so the analytic solution (u, v, ϕ) on the circle can be mapped to the kidney-shaped region without a singularity problem. The streamfunction is shown in Fig. 1 and u and v obtained by differentiating the streamfunction are shown in Fig. 2.

To construct a more realistic solution with small spatial scale and that evolves in time, we tested a streamfunction $\Psi(x, y, t) = \Psi_B + \Psi_E$ with Ψ_B the steady-state background flow constructed above and Ψ_E a 350-km eddy. The eddy is constructed on the circle first and then mapped onto the kidney-shaped region by using the modified cardioid mapping. On the unit circle the eddy has the form

$$\Psi_E = 0.05\exp\left(-\frac{(x - x_0)^2 + (y - y_0)^2}{0.003(1 + \cos^2(0.2\pi t))}\right), \tag{3.7}$$

where

$$x_0 = -0.1 + 0.7\cos(0.4\pi t),$$
$$y_0 = 0.7\cos(0.4\pi t).$$

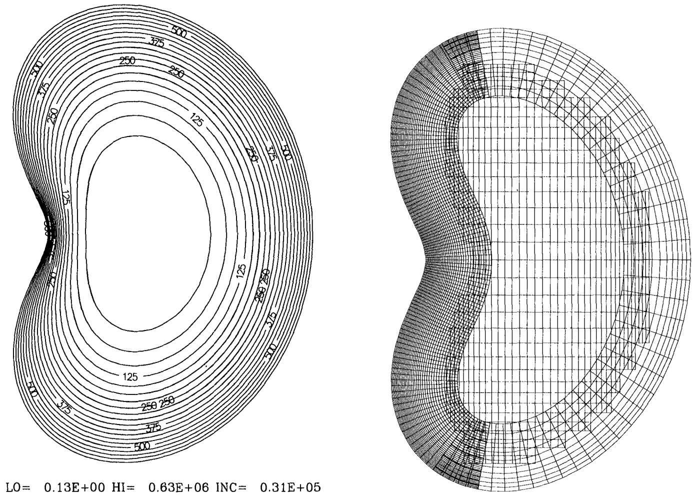


FIG. 1. *The streamfunction and the mesh on the kidney-shaped region.*

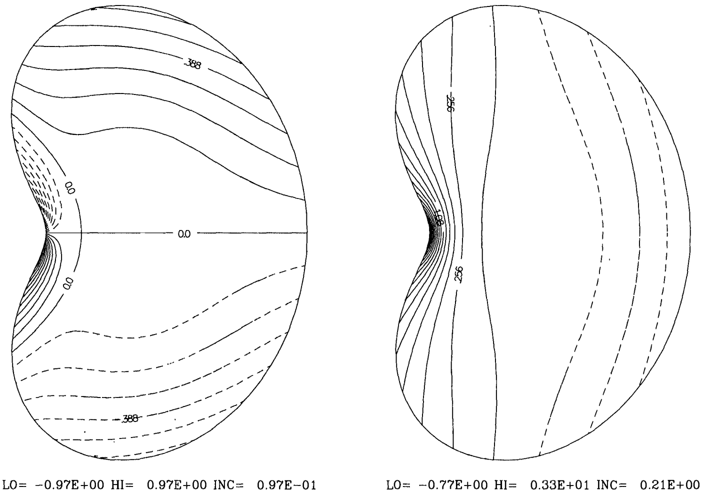


FIG. 2. *The contour plot of u and v on the kidney-shaped region.*

This eddy's radius is increased as it moves away from the west boundary. The streamfunction $\Psi_B + \Psi_E$ at $t = 0$ is shown in Fig. 3.

From the solution for the circle, it can be seen that the amplitudes of u and v are approximately 1 m/s near the boundary and ϕ is about 64 m^2/s^2 for $f = 10^{-4}$. This is also true for the transformed solution on the kidney-shaped domain. Thus, in the u and v equations of the reduced or full system, the truncation errors are not balanced. For example, in the u equation (2.1a), the truncation error of ϕ_x is approximately 64 times larger than the errors in the advection terms. This phenomenon is not particular to the special solution used here. Through scale analysis [2], it is known that the pressure gradient term is generally 10–100 times larger than the advection terms. This imbalance can be overcome in a number of ways. A straightforward solution is to use a higher-order finite difference scheme for ϕ_x. It is interesting to note that applying higher-order finite differences to all the terms in the shallow-water equations is not efficient because the truncation error in ϕ_x is dominant. So an efficient technique would use, for

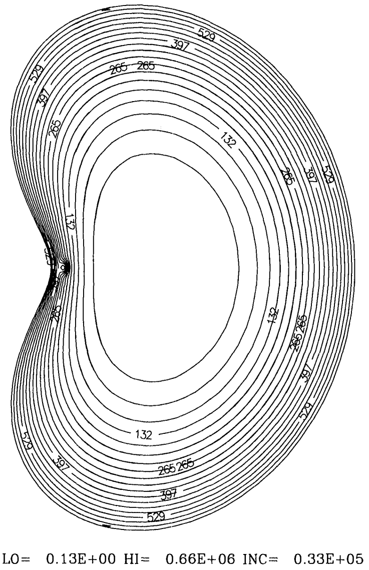


FIG. 3. *The streamfunction with the* 350-*km eddy.*

example, a fourth-order scheme for ϕ_x and a second-order scheme for the other terms. Treating fourth-order boundary conditions is another interesting issue in developing a well-posed system (see Henshaw [10]). In this paper, a change-of-variable approach is proposed for the two-dimensional reduced system and a similar idea can be applied to the three-dimensional reduced system. This approach can dramatically increase the accuracy without increasing either the resolution or the order of the finite difference scheme.

Suppose there is a function ϕ' satisfying

$$\phi = f\Psi + \phi'.$$

The amplitude of ϕ' is of the same order as u and v because the large part of ϕ is contained in the geostrophic portion $f\Psi$. Because the solution of the reduced system is nondivergent,

$$u = -\Psi_y, \quad v = \Psi_x,$$

the reduced system becomes

$$\begin{aligned} u_t + uu_x + vu_y + \phi'_x &= 0, \\ v_t + uv_x + vv_y + \phi'_y &= 0, \\ \nabla^2\phi' - 2\det[J(x, y; u, v)] &= 0. \end{aligned}$$

For this system, the truncation errors of the gradient of ϕ' and advective terms are the same size. This technique is called a change-of-variable method. Even for the two-dimensional flow, the original system requires a 3–10 times finer resolution for a second-order method in each direction to reach the same accuracy as the reduced system with a change-of-variable. In other words, the change-of-variable method is at least nine times more efficient than the system without the change-of-variable for two-dimensional flows and even more for three-dimensional ones. The original deviation of the geopotential ϕ can be recovered by solving the elliptic equation

$$\nabla^2\phi = f\zeta + \nabla^2\phi'$$

whenever it is needed. In the next section we discuss how to determine a unique solution from the boundary condition.

4. Composite grid method and the boundary treatment. The composite grid method was introduced over a decade ago [12], [13]. Chesshire and Henshaw [9] describe in considerable detail the implementation of this method for solving partial differential equations. This method allows multiple overlapping grids to cover an irregular domain of interest. Each grid is mapped onto a rectangular grid, for example, by a Coons [6] mapping. Then all numerical schemes can be applied as if on a rectangular grid. On the overlapping regions between grids, interpolation is used. Like other nested-grid and refinement methods (Ciment [5]), the stability of the composite grid method has been analyzed by Starius [12], [13]. Although Starius [13] considered only the Lax–Wendroff difference scheme, the same analysis can also be applied to other finite difference schemes. The nested-grid and refinement method have some difficulty dealing with an irregular boundary because it is complicated to make the separate components of the grid join precisely and smoothly. Usually, a stairwise nested grid is used to approximate an irregular basin, which reduces the accuracy near the boundary. The flexibility of the composite grid method for this problem can be seen in the grid construction for the kidney-shaped region in Fig. 1. For the kidney-shaped region, there are three overlapping grids: a western boundary, an eastern boundary, and an interior grid. The three overlapping grids cover the kidney-shaped domain to resolve the solution derived from the streamfunction shown in Fig. 1. Note that without the interior grid, there is a singularity problem when two grid lines intersect. Splines are used to fit irregular boundaries in general, and then the composite grid method handles the boundary smoothly and accurately. The boundary grids are tangential and orthogonal to the physical boundaries, and this makes the boundary conditions easy to implement. Note that much finer resolution is placed on the western boundary to help resolve the western boundary current.

To demonstrate the accuracy and efficiency of the composite grid method, the reduced shallow-water system is tested in the 4000-km wide by 6000-km high, kidney-shaped basin shown in Fig. 1 using the analytic solutions constructed in the previous section. Because the composite mesh has a fine resolution where Ψ changes quickly and where the eddy has a diameter of approximately 350 km before becoming large, and coarse resolution where Ψ changes slowly and the eddy is larger, it requires very few grid points to obtain a very accurate solution. This makes it possible to resolve this kind of solution on a workstation.

Solving the reduced system on a solid boundary basin requires explicit specification of the finite difference scheme and boundary conditions. In this paper, a simple finite different scheme, i.e., central finite differences in space and time, is used to discretize the reduced system, and biquadratic interpolation is used on the overlapped region of the composite grid. Near the boundary, the composite grid method provides a mapping from (x, y) to (r, s) such that the direction along r is the normal direction and the one along s is the tangential direction; i.e., (r, s) are orthogonal coordinates. This simplifies the implementation of the boundary conditions. Because

$$x = x(r, s), \qquad y = y(r, s), \tag{4.8}$$

the transformation between the velocity components is

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{\partial x}{\partial r}\frac{dr}{dt} + \frac{\partial x}{\partial s}\frac{ds}{dt} \\ \frac{\partial y}{\partial r}\frac{dr}{dt} + \frac{\partial y}{\partial s}\frac{ds}{dt} \end{pmatrix} = J(r, s; x, y)\begin{pmatrix} n \\ \tau \end{pmatrix} = J^{-1}(x, y; r, s)\begin{pmatrix} n \\ \tau \end{pmatrix}.$$

The mapping (4.8) or its inverse form,

$$r = r(x, y), \qquad s = s(x, y),$$

is chosen to be orthogonal; i.e.,

$$(r_x \; r_y)(s_x \; s_y)^T = 0.$$

The Jacobian matrix $J(x, y; r, s)$ is essentially orthogonal; i.e.,

$$J(x, y; r, s)J^T(x, y; r, s) = \Lambda(x, y),$$

where $\Lambda(x, y)$ is a diagonal matrix determined by the mapping

$$\Lambda(x, y) = \begin{pmatrix} r_x^2 + r_y^2 & 0 \\ 0 & s_x^2 + s_y^2 \end{pmatrix}.$$

Therefore, the time derivatives of (u, v) can be written as

$$(u_t \; v_t) = J^{-1}(x, y; r, s)(n_t \; \tau_t)^T$$

and for any function P the spatial derivatives can be written as

$$(P_x \; P_y)^T = J^T(x, y; r, s)(P_r \; P_s)^T = J^{-1}(x, y; r, s)\Lambda(x, y)(P_r \; P_s)^T.$$

The advective equations in the reduced system can be written as

$$\begin{aligned} &\begin{pmatrix} n_t \\ \tau_t \end{pmatrix} + J(x, y; r, s) \begin{pmatrix} nu_r + \tau u_s \\ nv_r + \tau v_s \end{pmatrix} \\ &+ \Lambda \begin{pmatrix} \phi_r \\ \phi_s \end{pmatrix} + fJ(x, y; r, s) \begin{pmatrix} -v \\ u \end{pmatrix} = 0 \end{aligned} \tag{4.9}$$

because

$$uu_x + vu_y = (u \quad v)(u_x \; u_y)^T = (n \quad \tau)J^{-T}(x, y; r, s)J^T(x, y; r, s)(u_r \; u_s)^T = nu_r + \tau u_s$$

and, similarly,

$$uv_x + vv_y = nv_r + \tau v_s.$$

For a solid-wall boundary condition, the normal velocity is zero, and τ can be updated by the second component of equation (4.9) by using only the function values of (u, v, ϕ) in the s-direction, i.e., along the boundary.

The first component of equation (4.9) gives a Neumann boundary condition for ϕ, i.e., the value of ϕ_r. Thus an elliptic equation with a Neumann boundary condition needs to be solved at every time step for the reduced shallow-water equations and the three-dimensional reduced system. That is, in general

$$\begin{aligned} \nabla^2\phi &= g_1(x, y, t), \\ \phi_r &= g_2(x, y, t) \quad \text{on the boundary,} \end{aligned}$$

where ϕ_r is the derivative in normal direction and $g_{1,2}$ are given functions of (x, y, t). In the following the direct solver of this Neumann boundary problem is discussed. How to implement the direct method for solving the elliptic equation with a Neumann boundary condition has a significant impact on applying the composite grid method to reduced systems. In addition to the fact that the direct method is faster than the multigrid method for low resolutions, all other techniques for solving the elliptic system rely on a proper implementation of the direct method. For example, the multigrid method needs a proper direct solver for its coarsest resolution and a conjugate gradient solver requires a proper projection of the Neumann boundary problem to obtain a nonsingular coefficient matrix. Therefore, it is crucial to implement the Neumann boundary problem accurately and efficiently for a direct solver.

The finite difference approximation of the continuous equations and the boundary condition gives a linear system

$$A\tilde{\phi} = b,$$

where $\tilde{\phi}$ is some enumeration of the grid function describing ϕ and b is the discrete right-hand side. For the Neumann boundary condition problem, it is known that A is rank-one deficient with a right-hand side null vector $r = (1, 1, \dots, 1)^T$, i.e., $Ar = 0$. Because $Null(A^T) \oplus \text{Range}(A) = R^n$, suppose $b = b_1 + b_2$, where $b_1 \in Null(A^T)$ and $b_2 \in \text{Range}(A)$, where $Null(A^T) = \{x | A^T x = 0\}$, the null space of A^T, and $\text{Range}(A) = \{x | \exists y \text{ s.t. } Ay = x\}$, the range space of A. The component b_1 may not be zero in general but of the order of the truncation error of the finite difference scheme and for second-order finite differences, $b_1 = O(\Delta x^2)$. An ideal solution to this problem is to find a minimal norm solution of

$$\min_{\tilde{\phi}} \|A\tilde{\phi} - b\|_2. \tag{4.10}$$

The problem is equivalent to solving

$$\begin{pmatrix} A & l \\ r^T & 0 \end{pmatrix} \begin{pmatrix} \tilde{\phi} \\ \tilde{x} \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}, \tag{4.11}$$

where l and r are the left and right null vectors of A; i.e., $l^T A = Ar = 0$ and usually the coefficient matrix is called the augmented matrix of A, which is nonsingular. One additional computation is that of the left null vector because the right one is known. If the direct solver is used, an LU decomposition is usually computed, for example, by the Yale sparse matrix solver (YSMP) [8] or the Harwell package (MA28) [7]. (Symmetric sparse decomposers are not suitable because the matrix is not symmetric if interpolation is used in the composite grid method.) The null vector l can be computed inexpensively. Suppose

$$A = P_l L U P_r, \tag{4.12}$$

where P_l and P_r are the pivot matrices (if only the row pivot is considered in the decomposition, $P_r = I$), and

$$U = \begin{pmatrix} U_1 & U_2 \\ 0 & u_{nn} \end{pmatrix},$$

where u_{nn} is zero theoretically and of round-off error size numerically because of the singularity of A. There are two ways to compute l. One is to solve $l = P_l^{-T} L^{-T} e_n$, where $e_n = (0, 0, \dots, 0, 1)^T$, and the computation of l requires only one backsubstitution. In this paper, another way, the inverse iteration method, is used to compute l because one does not need to have any knowledge about the data structure of MA28 or YSMP storing the L and P matrices. The method is very efficient because one iteration is sufficiently accurate for the numerical approximation.

Clearly (4.11) contains dense row and column vectors, namely, l and r. If an LU decomposer is applied directly to the augmented matrix, the fill-ins may be significant. The decomposition of the augmented matrix and the decomposition of A are tested on the circular basin with the mesh shown in Fig. 4. The former decomposition takes eight times longer than the latter. One can modify the pivoting such that the last row and column of the augmented matrices will not be pivoted until the last row and column of matrix A are reached. This requires some knowledge of the data structure of a matrix solver. In this paper, the LU decomposition

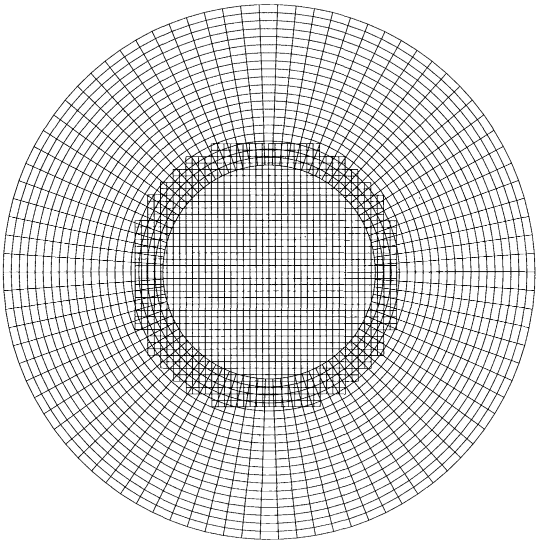


FIG. 4. *The composite grid on the circle.*

is applied to A. By using the computed l, one can obtain the minimal norm solution of (4.10) by solving

$$P_l L U P_r \tilde{\phi} = b - \frac{l^T b}{l^T l} l \tag{4.13}$$

and setting the last component of $P_r \tilde{\phi}$ to zero, which projects $\tilde{\phi}$ in the range space of A and ensures that $\tilde{\phi}$ has a minimal norm. For simplicity, let $P_l = P_r = I$. Because $l = L^{-T} e_n$, the last component of

$$L^{-1}\left(b - \frac{l^T b}{l^T l} l\right)$$

is

$$e_n^T L^{-1} b - \frac{e_n^T L^{-1} b}{\|L^{-T} e_n\|^2} e_n^T L^{-1} L^{-T} e_n = 0.$$

Thus (4.13) exactly holds. For the solution of (4.10), $\|A\tilde{\phi} - b\| = |l^T b| / \|l\|$, i.e., $\|A\tilde{\phi} - b\| = \min \|A\phi - b\|$. This method is implemented in this paper to test the numerical approximation of the reduced system.

5. Numerical results.

In this section, numerical results are presented. On the circular domain, results with and without the change-of-variable technique are compared. Following the numerical tests for the steady-state solution on the circular region, the numerical results for the untransformed, forced, reduced shallow-water equations on the kidney-shaped region are presented. Finally, the mesoscale eddy is considered.

The numerical results presented use the l_2-norm to calculate the relative errors and all the numerical errors are plotted in time for two weeks for all resolutions for both the circular and kidney-shaped regions.

5.1. Steady-state solution on the circular region and divergence damping technique.

In this subsection, numerical results are presented for the circular region because the analytic solution for the circular region does not require forcing terms.

TABLE 1

The relative errors in u on the circular region on day 14 *for low* (357 *km*), *medium* (178 *km*), *and high* (89 *km*) *resolutions* (%).

Different systems	Low	Medium	High
Original system	inf	128.3	44.5
System with damping	9.25	6.76	3.53
System with change-of-variable	0.98	0.25	0.06

Consider the analytic solution given by (3.6). The initial values are obtained using (3.6). Three resolutions are tested in this subsection and they have 357-km (low), 178-km (medium), and 89-km (high) minimum grid sizes, respectively. (Figure 4 shows the medium resolution grid.) Because a second-order finite difference is used in this paper, the relative errors should be reduced by a factor of four when the resolution is doubled (i.e., the grid size is halved). On the first time step, the relative errors of the three resolutions in u are 0.75×10^{-2}, 0.21×10^{-2}, and 0.49×10^{-3} (sizes are similar in v and ϕ). Although the errors do not reduce exactly by a factor of four from the low to the medium resolution because the low resolution may not be in the asymptotic range, the errors reduce properly from the medium to the high resolution.

Table 1 shows the increases with time of the relative errors for the three resolutions. It is found that the residual of (2.2c) grows quickly; i.e., the divergence $\delta = u_x + v_y$ grows too fast. To control the divergence, consider the divergence equation derived from (2.2a) and (2.2b):

$$\delta_t + \delta^2 + \nabla^2\phi - f\zeta - 2\det[J(x, y; u, v)] = 0.$$

The reduced shallow-water equations solve the elliptic equation (2.3) to obtain ϕ with the assumption $\delta = 0$. However, δ is not zero numerically but can be forced to be zero at the next time step in the following manner:

$$\frac{0 - \delta^{n-1}}{2\Delta t} + \nabla^2\phi - f\zeta - 2\det[J(x, y; u, v)] = 0,$$

where the low-order term δ^2 has been ignored. The first term is usually called a damping mechanism. This mechanism is used to test the numerical approximation of the reduced shallow-water equations, and the relative errors of u are shown in Table 1 using the same time step for all three resolutions. When the change-of-variable technique discussed in §3 is used, the accuracy can be increased dramatically and the damping mechanism is unnecessary for the circular case. Table 1 shows the increases of the relative errors in time for the three resolutions with the change-of-variable technique. In a two-month integration, even using the low resolution, the relative error in u is 4.8%.

The numerical approximation for the reduced system is very efficient. The Sun SS-2 central processing unit time used for a two-week integration for the three resolutions was as follows: low, 1.4 minutes; medium, 3.2 minutes; and high, 11.8 minutes. Note the time indicated does not include the preprocessing time (e.g., the time for constructing the grids and decomposition of the matrices).

5.2. Steady-state solution on the kidney-shaped region. In this subsection we test the solution for the untransformed, forced, reduced shallow-water system. Two resolutions are tested. Figure 1 shows the high resolution and the low resolution is obtained by halving the grid size of the high one. For the test purpose, ϕ is chosen as $\phi = f\Psi$. With the change-of-variable technique, the relative errors of the low and high resolutions in u are 0.65×10^{-3} and 0.19×10^{-3} without the damping mechanism. Table 2 shows the increase of the relative errors in time. After two months, the low resolution can achieve relative errors under 10%.

TABLE 2
Relative errors in u on the Kedney domain on day 14 *for the low and high resolutions (%).*

Different solutions	Low	High
Steady flow	3.67	0.90
With eddy	inf	57.4
Eddy passing overlap	127.5	20.6

5.3. A mesoscale eddy on the kidney-shaped region. It is difficult to resolve the eddy constructed in §3. Even with the change-of-variable technique, the relative errors are large. More resolution could be placed on the western boundary to resolve the eddy when no change-of-variable is made, but then the memory required would exceed the capacity of the SS-2 workstation. Therefore, we present only the numerical results from the change-of-variable technique and the damping mechanism. The eddy takes five months to travel a circle. Table 2 shows the relative errors in u without the background flow. The eddy starts at the position shown in Fig. 3 and stops two weeks later. From Table 2, it can be seen that the relative error can be around 10% if the resolution on the western boundary shown in Fig. 1 is doubled. Several tests have been made to allow the eddy to pass through the interpolation regions (i.e., the test starts near the overlapping areas and the eddy passes the areas in two weeks), and the relative errors are smaller than the errors shown in Table 2 because the eddy becomes larger in the overlapping areas than in the center of the western boundary. For example, Table 2 shows the relative errors of u for the two resolutions in two weeks when the eddy passes the overlapping area between the western and eastern boundaries.

6. Conclusions. The composite grid method has been applied to the reduced system for the shallow-water equations. From the analyses and numerical experiments in this paper, this combination of the reduced system with the composite grid method has been shown to be efficient and accurate, particularly for irregular boundary basins.

Acknowledgments. The authors would like to thank Dezso Devenyi for his careful and constructive comments during the internal review and Christy Sweet for her editorial review.

REFERENCES

[1] G. L. Browning, J. J. Hack, and P. N. Swarztrauber, *A comparison of three numerical methods for solving differential equations on the sphere*, Monthly Weather Rev., 117 (1989), pp. 1058–1075.

[2] G. L. Browning, W. R. Holland, and H.-O. Kreiss, *A comparison of differential systems and numerical methods for the computation of smooth oceanographic flows*, Dynamics Atmospheres Oceans, 16 (1992), pp. 499–526.

[3] G. L. Browning, W. R. Holland, H.-O. Kreiss, and S. J. Worley, *An accurate hyperbolic system for approximately hydrostatic and incompressible oceanographic flows*, Dynamics Atmospheres Oceans, 14 (1990), pp. 303–332.

[4] G. L. Browning, A. Kasahara, and H.-O. Kreiss, *Initialization of the primitive equations by the bounded derivative method*, J. Atmospheric Sci., 37 (1980), pp. 1424–1436.

[5] M. Ciment, *Stable difference schemes with uneven mesh spacing*, Math. Comp., 25 (1971), pp. 219–227.

[6] S. A. Coons, *Surfaces for Computer Aided Design of Space Forms*, Thesis, Dept. of Mech. Engrg., Massachusetts Inst. of Tech., Cambridge, 1964.

[7] I. S. Duff, MA28—*A Set of Fortran Subroutines for Sparse Unsymmetric Linear Equations*, Report AERE R8730, HMSO, London, 1977.

[8] S. C. Eisentat, M. C. Gursky, M. H. Schultz, and A. H. Sherman, *Yale sparse matrix package.* 1*: The symmetric codes*, Internat. J. Numer. Methods Engrg., 18 (1982), pp. 1145–1151.

[9] G. Chesshire and W. D. Henshaw, *Composite overlapping meshes for the solution of partial differential equations*, J. Comput. Phys., 90 (1990), pp. 1–64.

[10] W. D. Henshaw, *A fourth-order accurate method for the incompressible Navier-Stokes equations on overlapping grids*, J. Comput. Physics, 113 (1994), pp. 13–15.

[11] H.-O. Kreiss, *Problems with different time scales for partial differential equations*, Comm. Pure Appl. Math., 33 (1980), pp. 399–440.

[12] G. Starius, *Composite mesh difference method for elliptic boundary value problems*, Numer. Math. 28 (1977), pp. 243–258.

[13] ———, *On composite mesh difference method for hyperbolic differential equations*, Numer. Math. 35 (1980), pp. 241–255.

SIAM J. SCI. COMPUT.
Vol. 17, No. 5, pp. 1135–1149, September 1996

007

A SPARSE APPROXIMATE INVERSE PRECONDITIONER FOR THE CONJUGATE GRADIENT METHOD[*]

MICHELE BENZI[†], CARL D. MEYER[‡], AND MIROSLAV TŮMA[§]

Abstract. A method for computing a sparse incomplete factorization of the inverse of a symmetric positive definite matrix A is developed, and the resulting factorized sparse approximate inverse is used as an explicit preconditioner for conjugate gradient calculations. It is proved that in exact arithmetic the preconditioner is well defined if A is an H-matrix. The results of numerical experiments are presented.

Key words. sparse approximate inverses, preconditioned conjugate gradient method, H-matrices, incomplete factorizations

AMS subject classifications. 65F10, 65F35, 65F50, 65Y05

1. Introduction.

In this paper we develop a method for computing an incomplete factorization of the inverse of a symmetric positive definite (SPD) matrix A. The resulting factorized sparse approximate inverse is used as an explicit preconditioner for the solution of $Ax = b$ by the preconditioned conjugate gradient (PCG) method. Due to the fact that an explicit preconditioning step only requires matrix–vector products, explicit preconditioners are of considerable interest for use on parallel computers [7, 9, 17, 19–21]. This is in contrast with more traditional preconditioners based on incomplete factorizations of the coefficient matrix A which necessitate triangular solves (a serial bottleneck) in the preconditioning steps. Sparse incomplete inverses are also useful in the construction of sparse approximate Schur complements for use in incomplete block factorization preconditioners [2, 8]. Furthermore, our preconditioner does not require that A be explicitly stored, a feature which is useful for some problems where A is only implicitly given as an operator.

The paper is organized as follows. In §2 we describe the main idea upon which the preconditioner is based. Section 3 is devoted to a proof of the existence of the incomplete inverse factorization for H-matrices, while in §§4 and 5 implementation details and the results of numerical experiments are discussed. Our experiments indicate that this preconditioning strategy can insure rapid convergence of the PCG iteration with convergence rates comparable with those of the best serial preconditioners. In §6 we draw some conclusions and we indicate some future research directions.

This paper can be viewed as a natural outgrowth of work on a direct sparse linear solver based on oblique projections [3, 4, 28].

[*] Received by the editors July 20, 1994; accepted for publication (in revised form) April 21, 1995.

[†] Dipartimento di Matematica, Università degli Studi di Bologna, 40127 Bologna, Italy and CERFACS, 42 Ave. G. Coriolis, 31057 Toulouse Cedex, France (benzi@dm.unibo.it and benzi@cerfacs.fr).

[‡] Mathematics Department, North Carolina State University, Raleigh, NC 27695-8205 (meyer@math.ncsu.edu). The work of this author was supported in part by National Science Foundation grants DMS-9403224 and CCR-9413309 and by computing grants from the North Carolina Super Computing Center.

[§] Academy of Sciences of the Czech Republic, Institute of Computer Science, 182 07 Prague 8–Libeň, Czech Republic (tuma@uivt.cas.cz). The work of this author was supported in part by grants GA CR 201/93/0067 and GA AS CR 230401.

2. Computing an incomplete inverse factorization. If $A_{n\times n}$ is an SPD matrix, then a factorization of A^{-1} can readily be obtained from a set of conjugate directions $z_1, z_2, \ldots, z_n$ for A. If

$$Z = [z_1, z_2, \ldots, z_n]$$

is the matrix whose i th column is z_i, we have

$$Z^TAZ = D = \begin{pmatrix} p_1 & 0 & \cdots & 0 \\ 0 & p_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & p_n \end{pmatrix} \quad \text{where} \quad p_i = z_i^T A z_i.$$

It follows that

$$A^{-1} = ZD^{-1}Z^T,$$

and a factorization of A^{-1} is obtained. A set of conjugate directions z_i may be constructed by means of a "conjugate Gram–Schmidt" (or A-orthogonalization) process applied to any set of linearly independent vectors $v_1, v_2, \ldots, v_n$. The choice $v_i = e_i$ (the i th unit vector) is computationally convenient. The resulting Z matrix is unit upper triangular; indeed,

$$Z = L^{-T} \quad \text{where} \quad A = LDL^T$$

is the root-free Cholesky factorization of A. Denoting the i th row of A by a_i^T , the inverse factorization algorithm can be written as follows.

THE INVERSE FACTORIZATION ALGORITHM

```
(1) Let $z_i^{(0)} := e_i \quad (1 \le i \le n)$
(2) for $i = 1, 2, \ldots, n$
        for $j = i, i+1, \ldots, n$
            $p_j^{(i-1)} := a_i^T z_j^{(i-1)}$
        end
        if $i = n$ go to (3)
        for $j = i+1, \ldots, n$
            $z_j^{(i)} := z_j^{(i-1)} - \left(\frac{p_j^{(i-1)}}{p_i^{(i-1)}}\right) z_i^{(i-1)}$
        end
    end
(3) Let $z_i := z_i^{(i-1)}$ and $p_i := p_i^{(i-1)}$ , for $1 \le i \le n$ .
    Return $Z = [z_1, z_2, \ldots, z_n]$ and $D = \begin{pmatrix} p_1 & 0 & \cdots & 0 \\ 0 & p_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & p_n \end{pmatrix}$ .
```

Notice that the matrix A need not be explicitly stored—only the capability of forming inner products involving the rows of A is required. This is an attractive feature for cases where the matrix is only implicitly given as an operator. Once Z and D are available, the solution of $Ax = b$ can be computed as

$$x^* = A^{-1}b = ZD^{-1}Z^Tb = \sum_{i=1}^{n} \left(\frac{z_i^T b}{p_i} \right) z_i.$$

A similar algorithm was first proposed in [14]; see also [13, 18]. Further references and a few historical notes can be found in [3, 4]. For a dense matrix this method requires roughly twice as much work as Cholesky. For a sparse matrix the cost can be substantially reduced, but the method is still impractical because the resulting Z tends to be dense. The idea of computing a sparse approximation of Z to construct a preconditioner for the conjugate gradient method was first proposed in [3] (see also [4, 5]). This paper is devoted to developing and testing this idea.

Sparsity is preserved by reducing the amount of fill-in occurring in the computation of the z-vectors (that is, above the main diagonal in the unit upper triangular matrix Z). This can be achieved either by ignoring all fill outside selected positions in Z or by discarding fill whose magnitude falls below a preset drop tolerance (see §4 for details). The motivation for this approach is based upon theoretical results and computer experiments which show that many of the entries in the inverse (or in the inverse Cholesky factor) of a sparse SPD matrix are small in absolute value [2, 10, 25]. Several authors have exploited this fact to construct explicit preconditioners based on sparse approximate inverses [2, 20, 21]. However, the approach taken in this paper is quite different from the previous ones.

If the incomplete inverse factorization process is successfully completed, one obtains a unit upper triangular matrix $\bar{Z}$ and a diagonal matrix $\bar{D}$ with positive diagonal entries such that

$$M^{-1} := \bar{Z}\bar{D}^{-1}\bar{Z}^T \approx A^{-1}$$

is a factorized sparse approximate inverse of A. It is shown in the next section that such an incomplete inverse factorization of A exists (in exact arithmetic) for arbitrary values of the drop tolerance and for any choice of the sparsity pattern in $\bar{Z}$ when A is an H-matrix. For general SPD matrices the process may break down due to the occurrence of negative or zero pivots $\bar{p}_i$. Although numerical experiments show that this breakdown is not very likely to occur for reasonably well-conditioned problems, it is necessary to safeguard the computation of the approximate pivots against breakdown in order to obtain a robust procedure (see §4).

In this paper we limit ourselves to SPD matrices, but it is possible to apply the inverse factorization algorithm to arbitrary matrices. In exact arithmetic, the procedure can be carried out provided that all leading principal minors of A are nonzero [3]. The resulting Z and D matrices satisfy

$$AZ = LD$$

where L, a unit lower triangular matrix, is not explicitly computed. Hence, Z is the inverse of U in the LDU factorization of A. The application of such an implicit Gaussian elimination method to the solution of sparse linear systems has been investigated in [3, 4, 28].

3. Existence of the incomplete inverse factorization. The preconditioner based on the incomplete inverse factorization of A exhibits many analogies with the classical incomplete LDU factorization of Meijerink and van der Vorst [24]. These authors proved that such an incomplete factorization is well defined for arbitrary zero structures of the incomplete factors if A is an M-matrix. In other words, if A is a nonsingular M-matrix, then the incomplete factorization can be carried out (in exact arithmetic) and the computed pivots are strictly positive. Furthermore, the pivots in the incomplete factorization are no smaller than the pivots in the exact factorization. In [23], Manteuffel extended the existence of incomplete LDU factorizations to the class of H-matrices. Recall that $A = [a_{ij}]$ is an H-matrix if $\hat{A} = [\hat{a}_{ij}]$ is an M-matrix where

$$\hat{a}_{ij} = \begin{cases} -|a_{ij}| & \text{when } i \neq j, \\ a_{ii} & \text{when } i = j. \end{cases}$$

Note that a diagonally dominant matrix is an H-matrix.

This result means that if A is a symmetric H-matrix, then the incomplete Cholesky factorization always exists and it can be used to construct an SPD preconditioner for the conjugate gradient method. If A is a general (non-H) SPD matrix, the incomplete factorization may break down due to the occurrence of zero pivots, or the corresponding preconditioner may fail to be positive definite due to the presence of negative pivots.

The same turns out to be true for the incomplete inverse factorization described in the previous section. Here we prove that in exact arithmetic the inverse factorization algorithm given in §2 will never break down provided that A is an H-matrix. In the symmetric case this implies that the approximate inverse $\bar{Z}\bar{D}^{-1}\bar{Z}^T$ is positive definite, so it may be used as preconditioner for the conjugate gradient method. This fact was first proved for M-matrices in [3].

The proof runs as follows. First we show that the incomplete process will never break down if A is an M-matrix. This is a consequence of the fact that dropping a nonzero fill-in in the computation of a vector z_j at step i is equivalent to setting the corresponding entry of the ith row of A to zero. Because the off-diagonal entries of an M-matrix are nonpositive and the entries of $\bar{Z}$ are nonnegative, this shows that the pivots $\bar{p}_i$ produced by the inexact scheme are greater than or equal to the exact pivots p_i. Since these are strictly positive for an M-matrix, no breakdown can occur during the inexact inverse factorization scheme.

Subsequently we show that when A is an H-matrix, the pivots p_i computed by the inverse factorization scheme are no smaller than the pivots $\hat{p}_i$ corresponding to the associated M-matrix.

When combined, these two results will insure the stability of the incomplete procedure for H-matrices. For symmetric matrices this will mean that the factorized approximate inverse is positive definite and can be used as a preconditioner for the conjugate gradient method. However, symmetry is not required in our proof.

PROPOSITION 3.1. *Let A be an M-matrix and let p_i be the pivots produced by the inverse factorization algorithm. If $\bar{p}_i$ are the pivots computed by the incomplete inverse factorization algorithm with any preset zero pattern in the strictly upper triangular part of Z or any value of the drop tolerance, then*

$$\bar{p}_i \geq p_i > 0.$$

Proof. From the identity $AZ = LD$ and the fact that Z and L are unit triangular matrices it follows that the pivots p_i can be expressed in terms of the leading principal minors Δ_i of A as

$$p_i = \frac{\Delta_i}{\Delta_{i-1}} \quad (1 \leq i \leq n; \;\; \Delta_0 = 1).$$

Because A is an M-matrix, all its leading principal minors are positive and therefore $p_i > 0$ for all i. After $i-1$ steps of the inverse factorization scheme, the column vectors $z_j^{(i-1)}$ $(i \le j \le n)$ are available. Let $z_{kj}^{(i-1)}$ denote the kth entry of $z_j^{(i-1)}$. At step i of the inverse factorization scheme, the following are computed:

$$p_j^{(i-1)} = \sum_{l=1}^{i-1} a_{il} z_{lj}^{(i-1)} + a_{ij} \quad (i \le j \le n). \tag{3.1}$$

Suppose now that a sparsity pattern is imposed on the z-vectors, or that all fill-in in the z-vectors whose magnitude falls below a given drop tolerance is to be dropped. The modified z-vectors will be denoted by $\bar{z}_j^{(i-1)}$, and the pivots are now given by

$$\bar{p}_i^{(i-1)} = \sum_{l=1}^{i-1} a_{il} \bar{z}_{li}^{(i-1)} + a_{ii}. \tag{3.2}$$

We show by induction that $\bar{p}_i^{(i-1)} > 0$ for $1 \le i \le n$. We also show that $\bar{p}_j^{(i-1)} \le 0$ for $i+1 \le j \le n$ and that $\bar{z}_j^{(i-1)} \ge 0$ (componentwise) for $i \le j \le n$ for all i. For $i = 1$ the inequalities are obviously true. Now fix $i \ge 2$ and assume that $\bar{p}_{i-1}^{(i-2)} > 0$, $\bar{p}_j^{(i-2)} \le 0$ for $i \le j \le n$, and $\bar{z}_j^{(i-2)} \ge 0$ for $i-1 \le j \le n$. It follows that in the updates

$$\bar{z}_j^{(i-1)} := \bar{z}_j^{(i-2)} - \left(\frac{\bar{p}_j^{(i-2)}}{\bar{p}_{i-1}^{(i-2)}} \right) \bar{z}_{i-1}^{(i-2)}$$

nonpositive quantities are subtracted from nonnegative quantities. Therefore, even after dropping, no component of $\bar{z}_j^{(i-1)}$ can become negative. That is, $\bar{z}_j^{(i-1)} \ge 0$ for $i \le j \le n$. Using this inequalities and the fact that the off-diagonal entries of an M-matrix are nonpositive, we see from (3.1) that $\bar{p}_j^{(i-1)} \le 0$ for $i+1 \le j \le n$. Finally, it is clear from (3.2) that $\bar{p}_i^{(i-1)} \ge p_i^{(i-1)}$. Thus, the pivots cannot become smaller because of dropping. Since the exact pivots are positive, this proves that the incomplete inverse factorization process will not break down. □

We explicitly observe that when A is an M-matrix, our method is guaranteed to produce a nonnegative approximate inverse.

Now let A be an H-matrix, and apply the inverse factorization scheme to A as well as to the associated M-matrix $\hat{A}$. In the sequel, quantities with hats correspond to the associated process on $\hat{A}$. We need to compare pivots and z-vectors for the original process (on A) and for the associated process (on $\hat{A}$). To do this we also need to introduce intermediate quantities—denoted with tildes—which are constructed with entries from $\hat{A}$ and with pivots from A.

PROPOSITION 3.2. *Let A be an H-matrix and let $\hat{A}$ be the associated M-matrix. If p_i and $\hat{p}_i$ denote the pivots computed by the inverse factorization scheme applied to A and to $\hat{A}$, respectively, then $p_i \ge \hat{p}_i$. Furthermore, if $\bar{p}_i$ denote the pivots computed by the incomplete inverse factorization algorithm applied to A, then $\bar{p}_i \ge \hat{p}_i$.*

Proof. Consider the elements $z_{lj}^{(k)}$ of $z_j^{(k)}$ as rational functions

$$z_{lj}^{(k)} = F_{lj}^{(k)}(a_{11}, \ldots, a_{kn}, p_1, \ldots, p_k)$$

dependent on the elements of A and on the pivots $p_1, \ldots, p_k$. Likewise, we can consider the entries $\hat{z}_{lj}^{(k)}$ as rational functions depending on the entries in the first k rows of $\hat{A}$ and on the corresponding pivots $\hat{p}_1, \ldots, \hat{p}_k$. Let

$$\tilde{z}_{lj}^{(k)} = F_{lj}^{(k)}(\hat{a}_{11}, \ldots, \hat{a}_{kn}, p_1, \ldots, p_k)$$

be computed in the same way as $\hat{z}_{lj}^{(k)}$ using $p_1, \ldots, p_k$ instead of $\hat{p}_1, \ldots, \hat{p}_k$. In the following, it helps to think of the pivots p_i, $\hat{p}_i$ as parameters, ignoring their dependency on the entries of A and $\hat{A}$, respectively. In this way, the entries of $z_j^{(k)}$, $\hat{z}_j^{(k)}$, and $\tilde{z}_j^{(k)}$ can be regarded as polynomials in the entries of A and $\hat{A}$, respectively. We will prove that $p_i \geq \hat{p}_i$ using induction on i. We make the following inductive assumptions for all $k \leq i-1$.

$$p_k \geq \hat{p}_k, \tag{3.3}$$

$$\hat{z}_{lj}^{(k)} \geq \tilde{z}_{lj}^{(k)} \text{ for } l \leq j, j \geq i, \tag{3.4}$$

$$\tilde{z}_{lj}^{(k)} \text{ has all its terms nonnegative (as a polynomial).} \tag{3.5}$$

I. For $i = 1$ we have $p_1 = a_{11} = \hat{a}_{11} = \hat{p}_1 > 0$ and $\hat{z}_{lk}^{(1)} \geq \tilde{z}_{lk}^{(1)} \geq 0$.
II. Using (3.1) for $\hat{p}_i$ we get

$$\hat{p}_i = \sum_{l=1}^{i-1} \hat{a}_{il} \hat{z}_{li}^{(i-1)} + \hat{a}_{ii} \leq \sum_{l=1}^{i-1} \hat{a}_{il} \tilde{z}_{li}^{(i-1)} + \hat{a}_{ii}.$$

This inequality follows from the inductive assumption $\hat{z}_{li}^{(i-1)} \geq \tilde{z}_{li}^{(i-1)}$ and from the fact that the $\hat{a}_{il}$'s are nonpositive (being off-diagonal elements of the associated M-matrix).

Notice that corresponding terms in the expressions of $\tilde{z}_{li}^{(i-1)}$ and $z_{li}^{(i-1)}$ as polynomials have the same absolute value, so they can differ only by the sign. Using the defining identities for the H-matrix, i.e., $\hat{a}_{ii} = a_{ii}$ and $\hat{a}_{ik} = -|a_{ik}|$ for $i \neq k$, we get

$$\sum_{l=1}^{i-1} \hat{a}_{il} \tilde{z}_{li}^{(i-1)} + \hat{a}_{ii} \leq \sum_{l=1}^{i-1} a_{il} z_{li}^{(i-1)} + a_{ii} = p_i.$$

All terms in $\hat{a}_{il} \tilde{z}_{li}^{(i-1)}$ on the left-hand side, when considered as a polynomial in elements of $\hat{A}$, are nonpositive. On the right-hand side, some of the terms in the expression for $z_{li}^{(i-1)}$ can be positive. But corresponding terms of these polynomials have the same absolute value, so they differ only by the sign. Hence the inequality.

Using the updating formula—given in the inverse factorization algorithm—for $\hat{z}_j^{(i)}$ we have

$$\hat{z}_j^{(i)} = \hat{z}_j^{(i-1)} - \frac{\sum_{l=1}^{i-1} \hat{a}_{il} \hat{z}_{lj}^{(i-1)} + \hat{a}_{ij}}{\hat{p}_i} \hat{z}_i^{(i-1)}.$$

Since the off-diagonal elements of the M-matrix are nonpositive, and since $p_i \geq \hat{p}_i > 0$, we obtain

$$\hat{z}_j^{(i)} \geq \tilde{z}_j^{(i-1)} - \frac{\sum_{l=1}^{i-1} \hat{a}_{il} \tilde{z}_{lj}^{(i-1)} + \hat{a}_{ij}}{p_i} \tilde{z}_i^{(i-1)} \equiv \tilde{z}_j^{(i)}.$$

This follows from the set of inequalities

$$-\hat{a}_{il}\hat{z}_{lj}^{(i-1)} \geq -\hat{a}_{il}\tilde{z}_{lj}^{(i-1)},$$

$$\hat{z}_i^{(i-1)} \geq \tilde{z}_i^{(i-1)},$$

$$\hat{p}_i^{-1} \geq p_i^{-1}.$$

Assembling everything together we have

$$-\frac{\sum_{l=1}^{i-1}\hat{a}_{il}\hat{z}_{lj}^{(i-1)} + \hat{a}_{ij}}{\hat{p}_i}\hat{z}_i^{(i-1)} \geq -\frac{\sum_{l=1}^{i-1}\hat{a}_{il}\tilde{z}_{lj}^{(i-1)} + \hat{a}_{ij}}{p_i}\tilde{z}_i^{(i-1)}.$$

This inequality is added to the inequality from the assumption

$$\hat{z}_j^{(i-1)} \geq \tilde{z}_j^{(i-1)}$$

to arrive at

$$\hat{z}_j^{(i)} \geq \tilde{z}_j^{(i)}.$$

In addition, all the terms of $\tilde{z}_j^{(i)}$ are nonnegative.

Using the inductive assumption and defining identities for the H-matrix it can also be seen that (3.5) is true for $k = i$.

Concerning the second statement, it is easily seen that the same inequality for the pivots holds when the inverse factorization algorithm is applied to A incompletely, as the same argument for the polynomial terms can be applied even when some elements of $\hat{z}_j^{(i)}$ are set to zero. □

It follows from Propositions 3.1 and 3.2 that the incomplete inverse factorization process will never break down (in exact arithmetic) when A is an H-matrix. This is true for arbitrary zero patterns in the strictly upper triangular part of $\bar{Z}$ and for arbitrary choices of the drop tolerance.

The pivots produced by the incomplete inverse factorization of an H-matrix are no smaller than the pivots produced by the incomplete inverse factorization of the associated M-matrix. However, they are not necessarily larger than the pivots produced by the exact inverse factorization of A, contrary to what happens in the M-matrix case. For example, consider the H-matrix

$$\begin{pmatrix} 4 & -1 & -\epsilon \\ -1 & 4 & 1 \\ -\epsilon & 1 & 4 \end{pmatrix}.$$

If $0 < \epsilon < 1/4$, the incomplete inverse factorization algorithm with drop tolerance $Tol = 1/16$ returns a pivot $\bar{p}_3$ which is smaller than the pivot p_3 produced by the exact inverse factorization scheme. This is in perfect analogy with incomplete Cholesky factorizations (see [23, p. 479]).

If A is not an H-matrix, the incomplete inverse factorization algorithm may break down. For instance, applying the algorithm with a drop tolerance $Tol = 0.06$ to the SPD matrix

$$\begin{pmatrix} 2.00 & 0.40 & 0.10 \\ 0.40 & 1.08 & 2.00 \\ 0.10 & 2.00 & 3.96 \end{pmatrix}$$

results in $\bar{p}_3 = 0$ (a breakdown).

In finite precision computations, zero or negative pivots may occur even for H-matrices, due to round-off errors. Also, trouble can be expected in the presence of extremely small pivots. Indeed, this is one way for severe ill conditioning to manifest itself. Furthermore, there are many applications leading to SPD matrices which are not H-matrices—typically, finite element analysis. It is therefore desirable to incorporate some safeguard mechanism in the incomplete algorithm which guarantees that the computation of the preconditioner will run to completion and that it will always produce a symmetric positive definite approximate inverse factorization. Similar techniques have been implemented in connection with incomplete Cholesky factorization preconditioning and with approximate Hessian modifications [23, 16, 27].

4. Notes on implementation. We have implemented the PCG algorithm with our approximate inverse preconditioner—hereafter referred to as AINV—based on the inverse factorization algorithm of §2 as well as with a standard incomplete Cholesky (IC) preconditioner. The purpose of this comparison is to explore some characteristic algorithmic properties of the explicit preconditioner and to get a feeling for the convergence rate for the explicit PCG method as compared with one of the best implicit preconditioners.

We first describe the IC preconditioner used in our comparison. It was computed by a standard column algorithm with symbolic and numeric phases (see [15, 22]). During the decomposition we removed all the elements of the factor less than a prescribed drop tolerance Tol. Necessary working space was thus dominated by the size (number of nonzero entries) of the lower triangular factor of the IC preconditioner.

This decomposition, which could break down for general (non-H) matrices, was modified by a standard stabilization; see [16]. The algorithm insures that all diagonal elements of D in the LDL^T decomposition are strictly positive and the absolute values of the elements of L satisfy a uniform upper bound in order to preserve numerical stability and to prevent excessively large elements in the factors. We refer to [16] for details.

Computing the AINV preconditioner is a slightly more complicated process. In the Cholesky case we can use an elimination tree structure to minimize symbolic integer overhead and working storage. Due to the complicated rules governing fill-in in the AINV case, it is not clear how to realize an analogous symbolic process. Therefore, we used a submatrix type of algorithm which updates at each step all the remaining z-vectors by a rank-one modification. We adopted dynamic data structures similar to those used in submatrix formulations of sparse unsymmetric Gaussian elimination (see [11,28,29]). This requires the user to provide an estimate for the number of nonzeros allowed in the preconditioner. Additionally, some elbow space is needed for the data structures. In our implementation the elbow space was four times the estimated space for storing the nonzeros in the preconditioner—similar to the implementation of sparse Gaussian elimination in the widely used packages MA28 [11] and Y12M [29].

However, there are some differences in the use of such data structures in Gaussian elimination and in the AINV procedure. For instance, these data structures are used in AINV for the matrix Z but not for A, now stored in static data structures. During the AINV process, A is delivered into the cache by rows since we need at each step only one row of A. Recall that in some cases it may even be possible to avoid storage of A altogether—e.g., when a routine is available to compute the action of A on a vector (we did not take advantage of this option in our implementation).

The amount of fill-in created during the computation of the AINV preconditioner in most of the first steps is very small and thus the integer overhead and CPU time spent in these initial stages is very small. This is in contrast with sparse Gaussian elimination (as represented, for instance, by MA28), where the proportion of integer overhead and CPU time is distributed more uniformly over the algorithmic steps.

The sizes of the data structures in the AINV case which are necessary in the top level of the memory hierarchy (cache and registers) were found to be small, often much smaller than the size of the preconditioner. This fact can strongly influence performance, especially on workstation equipment. Nevertheless, working storage for the implementation of AINV is larger than for the implementation of IC.

Sparsity was preserved on the basis of value rather than on the positions of fill-in. For capturing the relevant entries in the inverse Cholesky factor of A, this is a better strategy than imposing a preset sparsity pattern on Z. Consistency suggested that drop tolerances be used with IC as well.

Skipping some z-vector updates in step (2) of the inverse factorization algorithm when the coefficients $p_j^{(i-1)}/p_i^{(i-1)}$ were in some sense "small" produced bad numerical results, so no skipping was done.

In the AINV case we also implemented an algorithmic modification to avoid breakdown for general SPD (non-H) matrices. When some computed diagonal element $\bar{p}_i$ was too small—in our case, less than $\sqrt{\epsilon_M}$ where ϵ_M is the machine precision—we replaced it by

$$\bar{p}_i \longleftarrow \max\{\sqrt{\epsilon_M},\ \mu\sigma\theta\} \tag{4.1}$$

where

$$\mu = 0.1 \text{ (a relaxation parameter)},$$
$$\sigma = \max_{i\le k\le n-1}\left\{p_k^{(i-1)}\right\},$$
$$\theta = \|z_i^{(i-1)}\|_\infty \neq 0.$$

The rule (4.1) was chosen to avoid breakdowns due to very small or negative diagonal elements $\bar{p}_i$. It also has the effect of constraining the growth of elements in the z-vectors. This avoids break downs, but, just as in the IC case, there is no guarantee that we will get a good preconditioner after this regularization.

5. Numerical experiments. The following experiments illustrate some properties of the two preconditioners when applied within the PCG algorithm to SPD matrices. Nine test matrices were taken from the Harwell–Boeing collection [12] and the remaining two were kindly provided by G. Zilli (Padua University).

All experiments were run on a SGI Crimson computer with RISC processor R4000. Codes were written in Fortran 77 and compiled with the optimization level –O4. CPU time was measured using the standard function *dtime*. We also experimented with the compiling option $-MIPS2$ which enables double word loads and stores. It is known that this can substantially enhance the performance of floating point arithmetic of some codes. This option led to slightly improved timings in only a few cases, mostly for ICCG. The timings reported in the tables are the best between those obtained with the two compiling options.

All matrices were rescaled by dividing their elements by their largest nonzero entry, but no preordering of their elements was used. The right-hand side of each system was computed using the solution vector composed of ones. The PCG iteration was terminated when the 2-norm of the unpreconditioned residual had been reduced to less than 10^{-9}. The matrices used in the experiments correspond to finite element approximations to problems in structural engineering (NOS3, NOS5, PADUA1, PADUA2), finite difference approximations to elliptic PDEs (NOS6, NOS7, GR3030) and to modelling of power system networks (BUS matrices).

TABLE 1
Behavior of the unpreconditioned CG algorithm.

test matrix	n	density	time	iterations
NOS3	960	8,402	1.76	266
NOS5	468	2,820	0.88	468*
NOS6	675	1,965	1.66	675*
NOS7	729	2,673	1.89	729*
494BUS	494	1,080	0.91	494*
662BUS	662	1,568	1.30	614
685BUS	685	1,967	1.44	556
1138BUS	1,138	2,596	3.83	1,138*
GR3030	900	4,322	0.28	45
PADUA1	812	3,135	3.06	812*
PADUA2	1,802	13,135	24.11	1,802*

The listings in Table 1 are for the unpreconditioned CG algorithm. Column 1 is the name of the test matrix; column 2 lists the size (n) of the matrix; column 3 (density) gives the number of nonzeros in the lower triangular part including the diagonal of the test matrix; column 4 (time) reports the execution time in seconds; and column 5 (iterations) gives the number of iterations. An asterisk (*) in column 5 indicates that the algorithm failed to converge after n steps using the above mentioned stopping criterion, and computations were terminated.

From the description of our implementation it can be expected that the CPU times for computing the two preconditioners IC and AINV will be different because the IC computation has very small integer overhead. Of course, this difference may become negligible if a sequence of linear systems with the same coefficient matrix (or a slightly modified one) and different right-hand sides has to be solved, since the time for computing the preconditioners is then only a small fraction of the time required for the overall computation.

Drop tolerances parameterize IC and AINV in different ways. That is, using the same Tol value will produce very different results in the two cases. It is preferable to compare the two preconditioners in terms of amount of fill-in rather than to compare two preconditioners obtained using the same value of Tol. The key role in the experiments is played by the fill-in allowed in the preconditioners. Allowing more fill-in results in less PCG iterations, though not always in less overall CPU time. The relation between preconditioner size (measured by fill-in) and number of PCG iterations for AINV and IC is one of the objectives of our comparison.

Table 2 lists the results of applying PCG with different sizes (measured by fill-in) of IC and AINV on the 675×675 Harwell–Boeing test matrix NOS6 which is derived from Poisson's equation in an L-shaped region with mixed boundary conditions. The timings in this table do not include the time required to compute the preconditioner.

TABLE 2
Behavior of PCG using IC versus AINV on H–B test matrix NOS6.

IC			AINV		
fill-in	**iterations**	**time**	**fill-in**	**iterations**	**time**
675	87	0.33	743	76	0.32
897	53	0.18	780	74	0.32
912	51	0.18	1,135	54	0.26
1,204	38	0.14	1,208	47	0.18
1,439	32	0.14	1,300	40	0.21
1,520	28	0.10	1,502	37	0.16
1,565	24	0.10	3,654	22	0.14
1,918	8	0.03	17,387	6	0.09

The results in Table 2 indicate that by using preconditioners of restricted size (obtained by adjusting the drop tolerances), the iteration counts as well as the timings for IC and AINV preconditioning are comparable, even in scalar mode allowing slightly more fill-in for the AINV preconditioner. For preconditioners of comparable size, slightly more iterations are needed by AINV preconditioning.

If we keep the size of the preconditioners "moderate," we usually decrease overall CPU time. What moderate means here is strongly problem and architecture (CPU, memory hierarchy) dependent.

The AINV method tends to generate more fill-in than IC, and for small drop tolerances the fill-in for AINV can be so high on some structured problems that we can no longer talk of sparse approximate inverse preconditioning thus making the comparison with IC not very meaningful (a preconditioner can be considered sparse if it contains about the same number of nonzeros as the original matrix or less). However, as discussed in [3, 4], problems having irregular sparsity patterns seem to be well suited for the AINV preconditioner because fill-in is often reasonably low.

TABLE 3
Iteration counts and timings for the IC preconditioner in PCG.

Test Matrix	**Fill-in**	**PCG steps**	**IC time**	**PCG time**	**Fill-in**	**PCG steps**	**IC time**	**PCG time**
NOS3	2,290	150	0.08	1.45	10,875	32	0.08	0.38
NOS5	744	76	0.04	0.25	1,967	57	0.05	0.21
NOS6	912	51	0.04	0.19	1,439	32	0.05	0.14
NOS7	902	51	0.09	0.21	938	40	0.09	0.14
494BUS	532	197	0.02	0.51	807	114	0.01	0.31
662BUS	694	159	0.02	0.58	1,090	103	0.02	0.42
685BUS	719	184	0.03	0.69	1,554	77	0.04	0.31
1138BUS	1,183	316	0.07	1.82	2,084	121	0.07	0.63
GR3030	900	45	0.07	0.28	4,322	26	0.06	0.22
PADUA1	1,228	104	0.06	0.54	1,644	70	0.06	0.45
PADUA2	5,305	143	0.23	2.71	7,777	84	0.20	1.65

Table 3 shows iteration counts and timings for PCG using the IC preconditioner, and Table 4 gives the same information for PCG using the AINV preconditioner. These tables also report the time required to compute each preconditioner. Again, the timings for PCG refer to the iterative part only. The IC and AINV preconditioners were computed with similar restricted sizes up to about the original number of nonzeros. Drop tolerances for IC were taken between 0.0001 and 0.01, and drop tolerances for AINV were in the range 0.1 to 0.6. For each test matrix two sparse preconditioners were computed—the first being very sparse while the second contains roughly the same number of nonzeros as the test matrix being used.

TABLE 4
Iteration counts and timings for the AINV preconditioner in PCG.

Test Matrix	Fill-in	PCG steps	AINV time	PCG time	Fill-in	PCG steps	AINV time	PCG time
NOS3	1,946	139	0.27	1.53	6,213	89	0.27	1.14
NOS5	889	87	0.09	0.29	2,086	67	0.12	0.30
NOS6	743	76	0.11	0.32	1,502	37	0.10	0.17
NOS7	903	55	0.07	0.28	2,727	30	0.13	0.19
494BUS	683	173	0.07	0.48	899	110	0.10	0.35
662BUS	893	147	0.13	0.68	1,008	125	0.10	0.56
685BUS	808	178	0.09	0.74	1,836	90	0.11	0.46
1138BUS	1,808	205	0.15	1.22	2,013	156	0.21	0.86
GR3030	900	45	0.12	0.29	13,541	26	0.35	0.37
PADUA1	1,274	63	0.14	0.38	1,459	48	0.14	0.31
PADUA2	5,269	159	0.25	3.14	11,095	80	0.42	1.92

Our results indicate that implicit and explicit sparse preconditioners can have similar behavior—even in the scalar case. The fact that a somewhat higher fill-in is required by the AINV preconditioner in order to achieve the same reduction in the number of PCG iterations as with IC is only natural, since in AINV we are approximating the inverse Cholesky factor (usually a dense matrix), whereas IC is a sparse approximation to the Cholesky factor L itself. If $\bar{L}$ is an incomplete Cholesky factor of A and $\bar{Z}$ is an incomplete inverse Cholesky factor, and if the amount of nonzeros in these two matrices is about the same, then one can expect that $\bar{L}^{-1}$ will be substantially denser than $\bar{Z}$.

In other words, for the same amount of fill-in in the preconditioners, IC yields a better approximation to A^{-1} than AINV. But this comes at a price—namely that two triangular solves are needed at each PCG iteration. On the other hand, the price to pay for the explicitness afforded by the AINV preconditioner is the increased size of the preconditioner, so, on a scalar computer, IC has a slight edge over AINV. However, the situation could be reversed in a parallel computing environment thanks to the explicit nature of AINV. This is a point which warrants further research, and no firm conclusion can be drawn until a parallel version of AINV-PCG has been actually implemented and compared with recent work on parallel solution of sparse triangular systems (see, for instance, [1]). In any event, we observe that even in scalar mode there are problems for which AINV is superior to IC—e.g., PADUA1.

It should be observed that all test matrices used are M-matrices except for NOS3, NOS5, PADUA1, and PADUA2, and these are not even H-matrices. In no case was safeguarding necessary during the computation of the AINV preconditioners, whereas in a few cases IC shifted positive pivots away from zero by a very small amount. This did not adversely affect the convergence of PCG.

6. Conclusions and future work. Our study involved a novel approach to approximate inverse preconditioning for conjugate gradient calculations. One interesting feature of this technique is the fact that the entries of the coefficient matrix A are not explicitly needed, which may be useful for problems where A is only implicitly given as an operator. It was proven that the computation of the preconditioner has the same robustness as standard IC factorization, and numerical evidence was given to make the point that the new preconditioner is competitive with IC—even in scalar mode. The results presented in this paper suggest that our approximate inverse preconditioner can be a useful tool for the solution of large sparse SPD linear systems on modern high-performance architectures.

Future research will focus on efficient parallel implementations and on the extension to unsymmetric problems. A sparse approximate inverse preconditioner for a nonsymmetric matrix A may be obtained by constructing a set of approximate A-biconjugate directions. This can be achieved by applying the inverse factorization algorithm to both A and A^T together with suitable sparsity-preserving strategies. The resulting factorized sparse approximate inverse, which is guaranteed to exist when A is an H-matrix, is an explicit preconditioner which can be used to enhance the convergence of conjugate gradient-like methods for the solution of $Ax = b$.

A different approach, applicable to general sparse matrices, is based on the normal equations $A^TAx = A^Tb$. The solution of this system by the preconditioned conjugate gradient method (PCGNR) is an effective strategy for problems which are unsymmetric and strongly indefinite; see [26, 29]. Also, the PCGNR method is attractive for solving large sparse linear least squares problems. Some of the most effective preconditioners for PCGNR are based on incomplete orthogonal factorizations of A and do not require explicitly forming the matrix A^TA. These procedures compute a sparse approximation to the upper triangular factor R in the QR decomposition of A. It is known that this approach is more robust than computing an incomplete Cholesky factorization of A^TA (notice that R is the transpose of the Cholesky factor of A^TA).

A natural idea is to compute an approximate inverse preconditioner for A^TA based on the inverse factorization scheme of §2. At step i of the algorithm the ith row of A^TA is computed and used and then discarded. The resulting $\bar{Z}$ is a sparse approximation to R^{-1}. The PCGNR scheme can be carried out free of triangular solves in the preconditioning steps. This approach was found to be effective for problems in which A^TA enjoys some form of diagonal dominance but in general was not competitive with more traditional schemes based on variants of the Gram–Schmidt orthogonalization process. More important, there exist other methods which can be used to compute R^{-1} directly from A. A description of some incomplete orthogonalization methods for approximating R and R^{-1}, together with the results of numerical experiments on a variety of general sparse matrices (including rectangular ones), can be found in [6].

REFERENCES

[1] F. L. Alvarado, A. Pothen, and R. Schreiber, *Highly parallel sparse triangular solution,* in

Graph Theory and Sparse Matrix Computations, IMA Vol. 56, A. George, J. R. Gilbert, and J. W. H. Liu, eds., Springer, New York, 1994, pp. 141–157.

[2] O. AXELSSON, *Iterative Solution Methods,* Cambridge University Press, Cambridge, 1994.

[3] M. BENZI, *A Direct Row-Projection Method For Sparse Linear Systems,* Ph.D. thesis, Department of Mathematics, North Carolina State University, Raleigh, NC, 1993.

[4] M. BENZI AND C. D. MEYER, *A direct projection method for sparse linear systems,* SIAM J. Sci. Comput., 16 (1995), pp. 1159–1176.

[5] ———*An explicit preconditioner for the conjugate gradient method,* Proceedings of the Cornelius Lanczos International Centenary Conference, J. D. Brown et al., eds., Society of Industrial and Applied Mathematics, Philadelphia, 1994, pp. 294–296.

[6] M. BENZI AND M. TŮMA, *A comparison of some preconditioning techniques for general sparse matrices,* in Proc. of the Second IMACS International Symposium on Iterative Methods in Linear Algebra, S. Margenov and P. Vassilevski, eds., 1995, to appear.

[7] E. CHOW AND Y. SAAD, *Approximate inverse preconditioners for general sparse matrices,* in Proc. of the Colorado Conference on Iterative Methods, April 5–9, 1994.

[8] P. CONCUS, G. H. GOLUB, AND G. MEURANT, *Block preconditioning for the conjugate gradient method,* SIAM J. Sci. Statis. Comput., 6 (1985), pp. 220–252.

[9] J. D. F. COSGROVE, J. C. DIAZ, AND A. GRIEWANK, *Approximate inverse preconditioning for sparse linear systems,* Internat. J. Computer Math., 44 (1992), pp. 91–110.

[10] S. DEMKO, W. F. MOSS, AND P. W. SMITH, *Decay rates for inverses of band matrices,* Math. Comp., 43 (1984), pp. 491–499.

[11] I. S. DUFF, MA28 - *A Set of Fortran Subroutines for Sparse Unsymmetric Linear Equations,* Harwell Report AERE - R.8730, Harwell Labortories, 1980.

[12] I. S. DUFF, R. G. GRIMES, AND J. G. LEWIS, *Sparse matrix test problems,* ACM Trans. Math. Software, 15 (1989), pp. 1–14.

[13] D. K. FADDEEV AND V. N. FADDEEVA, *Computational Methods of Linear Algebra,* W. H. Freeman and Co., San Francisco, London, 1963.

[14] L. FOX, H. D. HUSKEY, AND J. H. WILKINSON, *Notes on the solution of algebraic linear simultaneous equations,* Quart. J. Mech. Appl. Math., 1 (1948), pp. 149–173.

[15] A. GEORGE AND J. W. H. LIU, *Computer Solution of Large Sparse Positive Definite Systems,* Prentice-Hall, Englewood Cliffs, NJ, 1981.

[16] P. E. GILL, W. MURRAY, AND M. H. WRIGHT, *Practical Optimization,* Academic Press, London, 1981.

[17] M. GROTE AND H. SIMON, *Parallel preconditioning and approximate inverses on the Connection Machine,* in Proc. of the Sixth SIAM Conference on Parallel Processing for Scientific Computing, R. Sincovec et al., eds., Society of Industrial and Applied Mathematics, Philadelphia, 1993, pp. 519–523.

[18] A. S. HOUSEHOLDER, *The Theory of Matrices in Numerical Analysis,* Blaisdell Publishing Co., New York, 1964.

[19] T. HUCKLE AND M. GROTE, *A new approach to parallel preconditioning with sparse approximate inverses,* Manuscript SCCM-94-03, Scientific Computing and Computational Mathematics Program, Stanford University, Stanford, CA, May 1994.

[20] L. YU. KOLOTILINA AND A. YU. YEREMIN, *Factorized sparse approximate inverse preconditioning I. Theory,* SIAM J. Matrix Anal. Appl., 14 (1993), pp. 45–58.

[21] E. A. LIPITAKIS AND D. J. EVANS, *Explicit semi-direct methods based on approximate inverse matrix techniques for solving boundary-value problems on parallel processors,* Math. Comput. Simulation, 29 (1987), pp. 1–17.

[22] J. W. H. LIU, *The role of elimination trees in sparse factorization,* SIAM J. Matrix Anal. Appl., 11 (1990), pp. 134–172.

[23] T. A. MANTEUFFEL, *An incomplete factorization technique for positive definite linear systems,* Math. Comp., 34 (1980), pp. 473–497.

[24] J. A. MEIJERINK AND H. A. VAN DER VORST, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix,* Math. Comp., 31 (1977), pp. 148–162.

[25] G. MEURANT, *A review of the inverse of symmetric tridiagonal and block tridiagonal matrices,* SIAM J. Matrix Anal. Appl., 13 (1992), pp. 707–728.

[26] Y. SAAD, *Preconditioning techniques for nonsymmetric and indefinite linear systems,* J. Comput. Appl. Math., 24 (1988), pp. 89–105.

[27] R. B. SCHNABEL AND E. ESKOW, *A new modified Cholesky factorization,* SIAM J. Sci. Statist. Comput., 11 (1990), pp. 1136–1158.

[28] M. TŮMA, *Solving Sparse Unsymmetric Sets of Linear Equations Based on Implicit Gauss Projection,* Technical rep. 556, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague, April 1993.

[29] Z. ZLATEV, *Computational Methods for General Sparse Matrices,* Kluwer Academic Publishers, Dordrecht, 1991.

SIAM J. SCI. COMPUT.
Vol. 17, No. 5, pp. 1150–1166, September 1996

008

ACCURACY OF THE DISCRETE FOURIER TRANSFORM AND THE FAST FOURIER TRANSFORM*

JAMES C. SCHATZMAN†

Abstract. Fast Fourier transform (FFT)-based computations can be far more accurate than the slow transforms suggest. Discrete Fourier transforms computed through the FFT are far more accurate than slow transforms, and convolutions computed via FFT are far more accurate than the direct results. However, these results depend critically on the accuracy of the FFT software employed, which should generally be considered suspect. Popular recursions for fast computation of the sine/cosine table (or twiddle factors) are inaccurate due to inherent instability. Some analyses of these recursions that have appeared heretofore in print, suggesting stability, are incorrect. Even in higher dimensions, the FFT is remarkably stable.

Key words. fast Fourier transform (FFT), discrete Fourier transform (DFT)

AMS subject classifications. 65T20, 65Y25, 68Q25, 42A65, 42C10

1. Introduction.

The Fourier transform is one of the most important fundamental mathematical tools in existence today. While the discrete Fourier transform (DFT) and fast Fourier transform (FFT) are generally considered to be stable algorithms, reported quantifications of the stability have been inconsistent and confusing. Some analyses neglect the effect of errors in the coefficients (also called the sine/cosine table or twiddle factors), which turns out to be potentially the largest source of error.

Gentleman and Sande [4] report an analysis of FFT errors (for the Cooley–Tukey algorithm) in which the root mean square (RMS) relative error is

$$E_{RMS} = 1.06 \sum_{j=1}^{K} (2p_j)^{\frac{3}{2}} \epsilon, \tag{1}$$

where ϵ is the machine epsilon and $N = p_1 p_2 ... p_K$. For $N = 2^K$, and using the radix-2 algorithm, this becomes

$$E_{RMS} = 8.48 \log_2 N\epsilon. \tag{2}$$

Their corresponding formula for the slow DFT is

$$E_{RMS} = 1.06(2N)^{\frac{3}{2}}\epsilon. \tag{3}$$

The results of Gentleman and Sande are upper bounds. Our own *typical* results are quite different, being asymptotically of far better (smaller) order when the twiddle factors are accurate.

Kaneko and Liu [7] give an analysis for the Cooley–Tukey FFT with several types of input data sequences. Their results are rather complex, but their conclusion that errors in the twiddle factors have virtually no effect on the results (compare their Figures 4 and 7) is misleading.

Calvetti [2] gives an involved analysis for the Cooley–Tukey algorithm and the slow transform. She separates the effects of roundoff errors in addition and multiplication. The results are

*Received by the editors April 12, 1993; accepted for publication (in revised form) May 12, 1995.

†Department of Mathematics, University of Wyoming, P.O. Box 3036, Laramie, WY 82071 (jcs@uwyo.edu). This research was supported partly by a University of Wyoming Faculty Grant-in-Aid.

$$
\begin{aligned}
&E_{RMS} = \sqrt{\log_2 N}\,\sigma_a, && \text{addition errors, FFT;}\\
&E_{RMS} = \frac{1}{2}\sqrt{\log_2 N}\,\sigma_a, && \text{multiplication errors, FFT;}\\
&E_{RMS} = \frac{\sqrt{n-1}}{n}\,\sigma_m, && \text{addition errors, slow DFT;}\\
&E_{RMS} = \frac{1}{n}\,\sigma_m, && \text{multiplication errors, slow DFT;}
\end{aligned}
\tag{4}
$$

where σ_a, σ_m are the standard deviations of the assumed independent random errors in addition and multiplication. These errors are relative to the maximum norm of the input data. Calvetti claims, "For very small expected value of the relative error for addition and multiplication, the traditional [slow] algorithm will produce more accurate results." She further suggests, "The FFT can be considered more accurate than the TFT [slow DFT] only if the expected value of the relative error for addition is of the same size or larger than the expected value of the relative error for multiplication." I do not believe that the analysis for the slow algorithm is correct, nor is the general sense of Calvetti's conclusion (that the slow DFT is accurate compared to the FFT) correct. I do agree with Calvetti's formulas for FFT errors, in the case where the twiddle factors are accurate. Calvetti's paper gives a useful bibliography of earlier work.

According to Gottlieb and Orszag [5], "Transform methods normally give no appreciable amplification of roundoff errors. In fact, the evaluation of convolution-like sums using FFTs often gives results with much smaller roundoff error than would be obtained if the convolution sums were evaluated directly." See Van Loan [12] for a more recent publication on this topic.

Our own conclusions concerning accuracy are that the FFT is *remarkably* stable, when implemented properly. The FFT is vastly more accurate than the slow DFT. However, the FFT is very sensitive to accuracy of the internal sine/cosine table (twiddle factors). A popular technique for calculating the sine/cosine table by recursion is ill-advised for this reason.

2. The DFT.

2.1. Properties of the standard DFT. The square DFT might be written $\vec{c} = G\vec{x}$ or

$$c_k = \sum_{n=0}^{N-1} e^{-\imath\omega_k t_n} x_n, \tag{5}$$

where most commonly

$$\omega_k = k\Delta\omega, \qquad 0 \le k \le N\text{–}1, \tag{6}$$

$$t_n = n\Delta t, \qquad 0 \le n \le N\text{–}1, \tag{7}$$

$$\Delta\omega\Delta t = \frac{2\pi}{N}. \tag{8}$$

2.1.1. Numerical errors. Errors in the computation of the DFT are limited to errors in the approximations of the sine/cosine phase factors and roundoff errors in multiplication and addition. To aid in characterizing these errors, we define the machine ϵ in the usual way as the smallest positive number such that $1 + \epsilon$ is distinguishable from unity in the floating point representation employed. For tests with 32- and 64-bit IEEE floating point, we use the corresponding $\epsilon_{32} = 6 \cdot 10^{-8}$ and $\epsilon_{64} = 1.1 \cdot 10^{-16}$.

A useful exact characterization of expected error with floating point calculations is difficult or impossible, particularly in the case of addition, because of the nature of the floating point

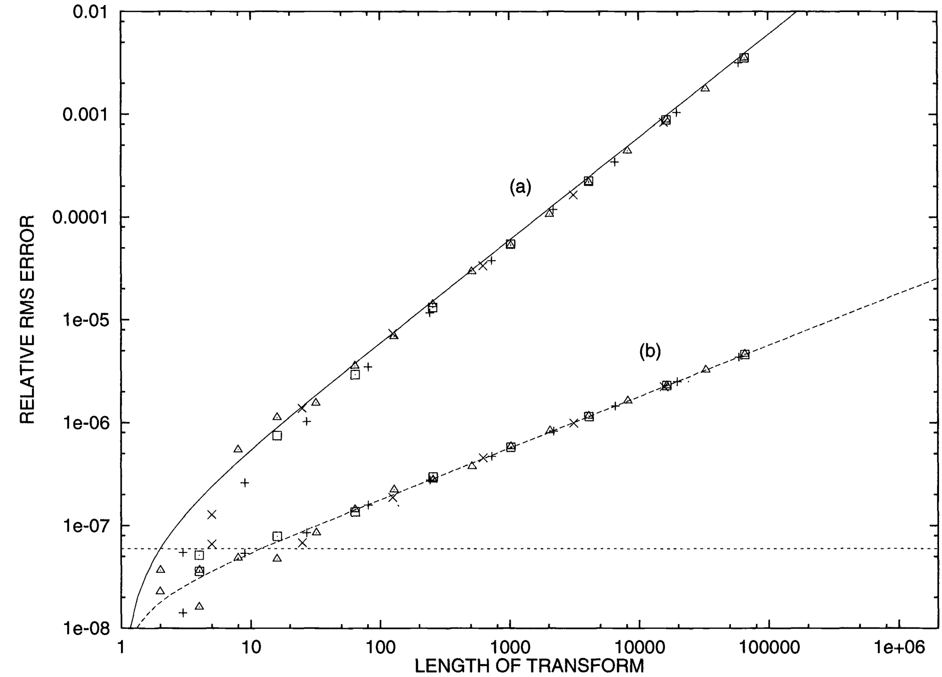


FIG. 1. *Relative RMS errors in the computed DFT for the* 32*-bit IEEE floating point slow Fourier transform using* (a) *the IBM* RS6000 *library* 32*-bit sine and cosine functions and the NCAR FFTPAK table calculation code, and* (b) *an accurate sine/cosine table.* △, □, +, *and* × *denote lengths that are powers of* 2, 3, 4, *and* 5, *respectively. The interpolating error functions (smooth curves) are* $e_1(N) = \epsilon_{32}(N{-}1)$ *and* $e_2(N) = 0.3\epsilon_{32}\sqrt{N{-}1}$. *The horizontal dashed line represents the constant* $\epsilon_{32} = 5.96 \cdot 10^{-8}$.

representation. Error bounds tend to lead to excessively pessimistic error estimates. Roughly, for data vectors exhibiting a reasonable degree of stationarity, if the errors associated with each coefficient, multiplication, and addition are uncorrelated (not a very reasonable assumption) the RMS error of the DFT would be expected to be approximately $\sqrt{N-1}$ times the error standard deviation associated with each term. If the errors are correlated, an RMS error proportional to $N - 1$ would be expected. Assuming that relative multiplication errors are uniformly distributed on $[-\epsilon/2, \epsilon/2]$ (a reasonable approximation if full rounding is used), the corresponding error standard deviation is $\epsilon/\sqrt{12}$. Assuming that relative addition errors are uniformly distributed on $[-\epsilon, \epsilon]$ (a poor approximation), the corresponding error standard deviation is $\epsilon/\sqrt{3}$.

Putting these ideas together, we obtain a rough estimate of the overall RMS error in the case of uncorrelated individual errors:

$$E_{RMS} = \sqrt{(N{-}1)\left(\frac{1}{12} + \frac{1}{3}\right)\epsilon^2 + (N{-}1)\sigma_C{}^2}, \tag{9}$$

where σ_C is the error standard deviation of the coefficients. If σ_C is $\epsilon/\sqrt{12}$, (9) reduces to $E_{RMS} = (\epsilon/\sqrt{2})\sqrt{N{-}1}$.

Figures 1 and 2 show a comparison of 128-bit floating point DFT results (taken to be truth) with

(1a) IEEE 32-bit DFT computations using the manufacturer's 32-bit sine/cosine functions,
(1b) IEEE 32-bit DFT computations modified to use an accurate table of phase factors,

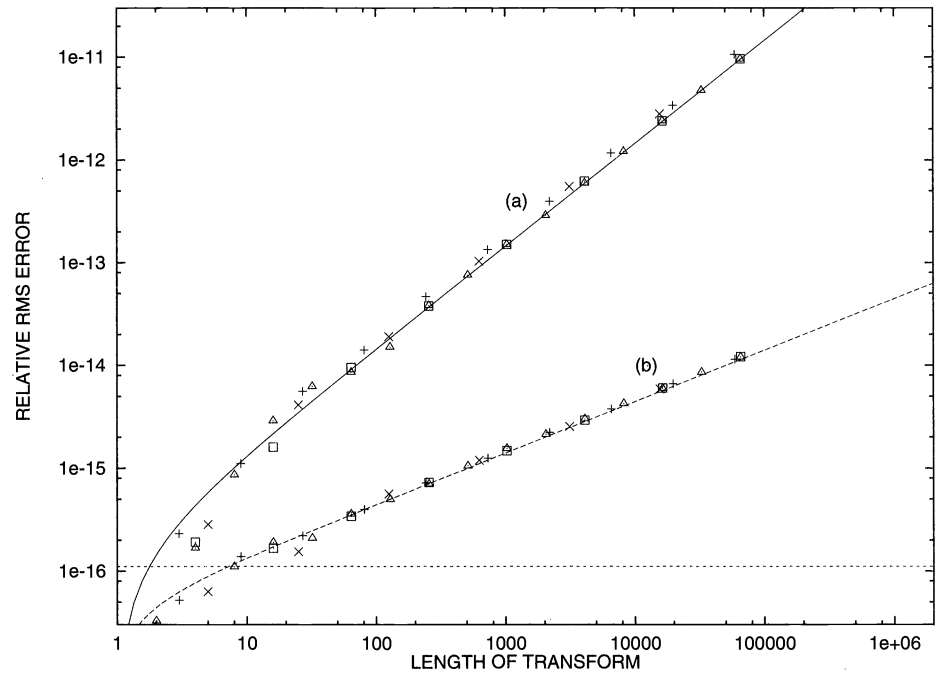


FIG. 2. *Relative RMS errors in the computed DFT for the* 64-*bit IEEE floating point slow Fourier transform using* (a) *the IBM RS6000 library* 64-*bit sine and cosine functions and the NCAR FFTPAK table calculation code, and* (b) *an accurate sine/cosine table.* △, +, □, *and* × *denote lengths that are powers of* 2, 3, 4, *and* 5, *respectively. The interpolating error functions (smooth curves) are* $e_1(N) = 1.3\epsilon_{64}(N{-}1)$ *and* $e_2(N) = 0.4\epsilon_{64}\sqrt{N{-}1}$. *The horizontal dashed line represents the constant* $\epsilon_{64} = 1.11 \cdot 10^{-16}$.

(2a) IEEE 64-bit DFT computations using the manufacturer's 64-bit sine/cosine functions,

(2b) IEEE 64-bit DFT computations modified to use an accurate table of phase factors. The tests were performed on an IBM RS/6000 computer. The input data was a series of independent, identically distributed Gaussian random sequences (for the real and imaginary components). The RMS difference between the two computations, scaled by the RMS value of the "true" result, is displayed as a function of transform length.

As shown in the figures, fits to the measured RMS errors were obtained for the following error functions:

(1a) 32-bit: $\epsilon_{32}(N{-}1)$,

(1b) improved 32-bit: $0.3\epsilon_{32}\sqrt{N{-}1}$,

(2a) 64-bit: $1.3\epsilon_{64}(N{-}1)$,

(2b) improved 64-bit: $0.4\epsilon_{64}\sqrt{N{-}1}$.

When accurate phase factors are used (Figures 1b and 2b) square root error growth is observed; results are obtained that are more accurate than predicted by (9) by about a factor of two. When inaccurate phase factors are used (Figures 1a and 2a), linear growth is observed, suggesting that individual errors are correlated.

From these results we conclude the following:

(1) Accuracy of the sine/cosine table is critical to overall performance of the numerical DFT. When the sine/cosine table of the FFT is inaccurate, errors in the DFT may be correlated and the overall RMS DFT errors are then roughly proportional to the length of the transform.

(2) The IBM RS/6000 library sine/cosine functions are not very accurate; we do not recommend their use except for noncritical applications. A similar observation has been made for the Cray CFT77 library routines. The computation of long slow DFTs with IEEE 32-bit floating point may be expected to be of questionable accuracy unless more accurate sine/cosine approximations are used. Using the RS/6000 library routines, only about five significant figures should be expected for about 100 points ranging down to two significant figures for about 100k points.

(3) When the sine/cosine table is accurate, performance is dramatically superior for both 32- and 64-bit computations, and the RMS DFT errors grow proportionally to the square root of the length of the transform. In this case, the DFT may be regarded as a stable process. However, there still may be significant error for very long transforms; five to six significant figures of accuracy may be expected for transforms in the 100k+ point range using 32-bit floating point.

There is obviously some sensitivity of the errors to the data. An extreme example is the case of data consisting of mostly zeros. The above results should be interpreted as typical rather than definitive.

3. The FFT.

3.1. Sources of error. It is helpful to list the theoretical sources of error:

(1) instability of the FFT computations associated with the factorization,

(2) instability of the underlying DFT blocks,

(3) errors in the sine/cosine table (twiddle factors),

(4) roundoff error in all of the computations, compounded randomly.

While all errors could be considered roundoff errors, the above breakdown is reasonable. In particular, the idea of approximating roundoff by an instability effect and a random effect is a powerful tool. The validity of this model for general numerical computations remains to be demonstrated; there is good agreement between theoretical and empirical results in this analysis.

We will show that sources (1) and (2) may be discounted (the associated computations are extremely stable). Sources (3) and (4) are the principal sources of error; in a properly designed code, source (4) will dominate.

3.2. FFT formulation. We formulate the conventional version of the FFT to concretize the differences between it and the slow DFT. Suppose N is even. We observe

$$c_k = \sum_{j=0}^{N-1} w^{kn} x_j = \sum_{j=0}^{\frac{N}{2}-1} w^{2kj} x_{2j} + w^k \sum_{j=0}^{\frac{N}{2}-1} w^{2kj} x_{2j+1}$$

$$(10) \qquad = \begin{cases} c_k^{(0)} + w^k c_k^{(1)}, & 0 \leq k \leq \frac{N}{2} - 1, \\ c_{k'}^{(0)} - w^{k'} c_{k'}^{(1)}, & \frac{N}{2} \leq k \leq N-1, \end{cases}$$

where $k' = k - \frac{N}{2}$, $w = e^{-\imath \frac{2\pi}{N}}$, $\vec{c}^{(0)}$ is the length $\frac{N}{2}$ DFT of the even terms $x_0, x_2, \ldots$, and $\vec{c}^{(1)}$ is the length $\frac{N}{2}$ DFT of the odd terms $x_1, x_3, \ldots$. This is the well-known result that a DFT of even length N can be computed as the combination of two DFTs of length $\frac{N}{2}$. The coefficients w^i and $w^{i'}$ are called the twiddle factors. In matrix notation (10) may be written

$$(11) \qquad \vec{c} = \begin{pmatrix} I_{\frac{N}{2}} & \Omega^{\frac{1}{2}}_{\frac{N}{2}} \\ I_{\frac{N}{2}} & -\Omega^{\frac{1}{2}}_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} \vec{c}^{(0)} \\ \vec{c}^{(1)} \end{pmatrix},$$

where

$$\Omega_q = \mathrm{diag}(z^0, z^1, z^2, \ldots, z^{q-1}), \qquad z = e^{-i\frac{2\pi}{q}}, \tag{12}$$

and $I_{\frac{N}{2}}$ is the $\frac{N}{2}$-by-$\frac{N}{2}$ identity matrix. Repeating this process for N a power of 2, we obtain

$$F_N = S_N^{(2)} \begin{pmatrix} S_{\frac{N}{2}}^{(2)} & \bigcirc \\ \bigcirc & S_{\frac{N}{2}}^{(2)} \end{pmatrix} \begin{pmatrix} S_{\frac{N}{4}}^{(2)} & \bigcirc & \bigcirc & \bigcirc \\ \bigcirc & S_{\frac{N}{4}}^{(2)} & \bigcirc & \bigcirc \\ \bigcirc & \bigcirc & S_{\frac{N}{4}}^{(2)} & \bigcirc \\ \bigcirc & \bigcirc & \bigcirc & S_{\frac{N}{4}}^{(2)} \end{pmatrix}$$

$$\cdots \begin{pmatrix} S_2^{(2)} & \cdots & \bigcirc \\ \vdots & \ddots & \vdots \\ \bigcirc & \cdots & S_2^{(2)} \end{pmatrix} P_N^{(2,2,2,\ldots,2)}, \tag{13}$$

where

$$\begin{cases} S_m^{(2)} = \begin{pmatrix} I_{\frac{m}{2}} & \Omega_{\frac{m}{2}}^{\frac{1}{2}} \\ I_{\frac{m}{2}} & -\Omega_{\frac{m}{2}}^{\frac{1}{2}} \end{pmatrix}, \\ P_N^{(2,2,2,\ldots,2)} = \text{bit-reversal ordering permutation matrix.} \end{cases} \tag{14}$$

For more general N, if $N = p_1 p_2 \cdots p_K$, where p_j are natural numbers, the FFT formula may be written

$$F_N = S_N^{(p_1)} \begin{pmatrix} S_{\frac{N}{p_1}}^{(p_2)} & & \bigcirc \\ & \ddots & \\ \bigcirc & & S_{\frac{N}{p_1}}^{(p_2)} \end{pmatrix} \begin{pmatrix} S_{\frac{N}{(p_1 p_2)}}^{(p_3)} & & \bigcirc \\ & \ddots & \\ \bigcirc & & S_{\frac{N}{(p_1 p_2)}}^{(p_3)} \end{pmatrix}$$

$$\cdots \begin{pmatrix} S_{p_K}^{(p_K)} & & \bigcirc \\ & \ddots & \\ \bigcirc & & S_{p_K}^{(p_K)} \end{pmatrix} P_N^{(p_1, p_2, \ldots, p_K)}, \tag{15}$$

where

$$S_m^{(p)} = \begin{pmatrix} (F_p)_{1,1} I_{\frac{m}{p}} & (F_p)_{1,2} \Omega_{\frac{m}{p}}^{\frac{1}{p}} & \cdots & (F_p)_{1,p} \Omega_{\frac{m}{p}}^{\frac{p-1}{p}} \\ (F_p)_{2,1} I_{\frac{m}{p}} & (F_p)_{2,2} \Omega_{\frac{m}{p}}^{\frac{1}{p}} & \cdots & (F_p)_{2,p} \Omega_{\frac{m}{p}}^{\frac{p-1}{p}} \\ \vdots & \vdots & & \vdots \\ (F_p)_{p,1} I_{\frac{m}{p}} & (F_p)_{p,2} \Omega_{\frac{m}{p}}^{\frac{1}{p}} & \cdots & (F_p)_{p,p} \Omega_{\frac{m}{p}}^{\frac{p-1}{p}} \end{pmatrix} \tag{16}$$

$$= F_m^{(p)} \, Q_m^{(p)}.$$

Here $F_m^{(p)}$ is the m-by-m matrix constructed from the p-by-p DFT matrix F_p by replacing each element of F_p by the product of the element and the m/p-by-m/p identity matrix. Also,

$$Q_m^{(p)} = \operatorname{diag}\left(I_{\frac{m}{p}}, \Omega_{\frac{m}{p}}^{\frac{1}{p}}, \Omega_{\frac{m}{p}}^{\frac{2}{p}}, \dots, \Omega_{\frac{m}{p}}^{\frac{p-1}{p}}\right) \tag{17}$$

is the diagonal matrix of twiddle factors, and $P_N^{(p_1,p_2,\dots,p_K)}$ is the mixed-radix digit-reversal ordering permutation matrix, defined as

$$\left\{\begin{aligned} P_N^{(p_1,p_2,\dots,p_K)}{}_{j,k} &= \delta_{j,k'}, \qquad 0 \le j,k \le N{-}1,\\ k' &= k'(k)\\ &= d_K + p_K(d_{K-1} + p_{K-1}(d_{K-2} + \cdots + p_2 d_1)),\\ k &= d_1 + p_1(d_2 + p_2(d_3 + p_3(d_4 + \cdots + p_{K-1}d_K))), \end{aligned}\right. \tag{18}$$

where the mixed-radix digits d_l satisfy $0 \le d_l \le p_l - 1$. The formula (15) represents the so-called mixed-radix time-decimation FFT algorithm.

A mathematically equivalent formula that leads to the frequency-decimation algorithm follows from the observation that

$$\begin{aligned} F_N &= \left(\tfrac{1}{N} F_N^*\right)^{-1}\\ &= N\left(P_N^{(p_1,p_2,\dots,p_K)}\right)^{-1} \begin{pmatrix} S_{p_K}^{(p_K)*} & & \bigcirc\\ & \ddots & \\ \bigcirc & & S_{p_K}^{(p_K)*} \end{pmatrix}^{-1}\\ &\qquad \cdots \begin{pmatrix} S^{(p_3)\ *}_{\frac{N}{(p_1p_2)}} & & \bigcirc\\ & \ddots & \\ \bigcirc & & S^{(p_3)\ *}_{\frac{N}{(p_1p_2)}} \end{pmatrix}^{-1}\\ &\qquad \begin{pmatrix} S^{(p_2)*}_{\frac{N}{p_1}} & & \bigcirc\\ & \ddots & \\ \bigcirc & & S^{(p_2)*}_{\frac{N}{p_1}} \end{pmatrix}^{-1} \left(S_N^{(p_1)*}\right)^{-1}. \end{aligned} \tag{19}$$

Then

$$\left(P_N^{(p_1,p_2,\dots,p_K)}\right)^{-1} = P_N^{(p_K,p_{K-1},\dots,p_1)} \tag{20}$$

and

$$\left(S_m^{(p)*}\right)^{-1} = \frac{1}{p} R_m^{(p)}, \tag{21}$$

where

$$R_m^{(p)} = \begin{pmatrix} (F_p)_{1,1} I_{\frac{m}{p}} & (F_p)_{1,2} I_{\frac{m}{p}} & \cdots & (F_p)_{1,p} I_{\frac{m}{p}} \\ (F_p)_{2,1} \Omega_{\frac{m}{p}}^{\frac{1}{p}} & (F_p)_{2,2} \Omega_{\frac{m}{p}}^{\frac{1}{p}} & \cdots & (F_p)_{2,p} \Omega_{\frac{m}{p}}^{\frac{1}{p}} \\ \vdots & \vdots & \vdots & \vdots \\ (F_p)_{p,1} \Omega_{\frac{m}{p}}^{\frac{p-1}{p}} & (F_p)_{p,2} \Omega_{\frac{m}{p}}^{\frac{p-1}{p}} & \cdots & (F_p)_{p,p} \Omega_{\frac{m}{p}}^{\frac{p-1}{p}} \end{pmatrix} \tag{22}$$

$$= Q_m^{(p)}\ F_m^{(p)}$$

so that

$$F_N = P_N^{(p_K, p_{K-1}, \dots, p_1)} \begin{pmatrix} R_{p_K}^{(p_K)} & & \bigcirc \\ & \ddots & \\ \bigcirc & & R_{p_K}^{(p_K)} \end{pmatrix} \tag{23}$$

$$\cdots \begin{pmatrix} R_{\frac{N}{(p_1 p_2)}}^{(p_3)} & & \bigcirc \\ & \ddots & \\ \bigcirc & & R_{\frac{N}{(p_1 p_2)}}^{(p_3)} \end{pmatrix} \begin{pmatrix} R_{\frac{N}{p_1}}^{(p_2)} & & \bigcirc \\ & \ddots & \\ \bigcirc & & R_{\frac{N}{p_1}}^{(p_2)} \end{pmatrix} R_N^{(p_1)}.$$

These formulas may be simplified by use of the Kronecker matrix product $\otimes$ which is defined so that $A \otimes B$ is the LI-by-MJ matrix

$$A \otimes B = \begin{pmatrix} A_{1,1}B & A_{1,2}B & \cdots & A_{1,M}B \\ A_{2,1}B & A_{2,2}B & \cdots & A_{2,M}B \\ \vdots & \vdots & & \vdots \\ A_{L,1}B & A_{L,2}B & \cdots & A_{L,M}B \end{pmatrix} \tag{24}$$

for an L-by-M matrix A and an I-by-J matrix B. Then

$$\begin{aligned} F_m^{(p)} &= F_p \otimes I_{\frac{m}{p}}, \\ S_m^{(p)} &= F_m^{(p)}\ Q_m^{(p)} \\ &= \left(F_p \otimes I_{\frac{m}{p}}\right)\ Q_m^{(p)}, \\ R_m^{(p)} &= Q_m^{(p)}\ F_m^{(p)} \\ &= Q_m^{(p)}\ \left(F_p \otimes I_{\frac{m}{p}}\right), \end{aligned} \tag{25}$$

and

$$\begin{aligned}
F_N &= S_N^{(p_1)} \left(I_{p_1} \otimes S^{(p_2)}_{\frac{N}{p_1}} \right) \left(I_{p_1 p_2} \otimes S^{(p_3)}_{\frac{N}{(p_1 p_2)}} \right) \\
&\quad \cdots \left(I_{p_1 p_2 \ldots p_{K-1}} \otimes S^{(p_K)}_{p_K} \right) P_N^{(p_1, p_2, \ldots, p_K)} \\
&= \left(F_{p_1} \otimes I_{\frac{N}{p_1}} \right) Q_N^{(p_1)} \left\{ I_{p_1} \otimes \left[\left(F_{p_2} \otimes I_{\frac{N}{p_1 p_2}} \right) Q^{(p_2)}_{\frac{N}{p_1}} \right] \right\} \\
&\quad \left\{ I_{p_1 p_2} \otimes \left[\left(F_{p_3} \otimes I_{\frac{N}{p_1 p_2 p_3}} \right) Q^{(p_3)}_{\frac{N}{(p_1 p_2)}} \right] \right\} \\
&\quad \cdots \left\{ I_{p_1 p_2 \ldots p_{K-1}} \otimes \left[F_{p_K} \; Q^{(p_K)}_{p_K} \right] \right\} \; P_N^{(p_1, p_2, \ldots, p_K)} \\
&= \left\{ I_{q_1} \otimes \left[\left(F_{p_1} \otimes I_{m_2} \right) Q^{(p_1)}_{m_1} \right] \right\} \\
&\quad \left\{ I_{q_2} \otimes \left[\left(F_{p_2} \otimes I_{m_3} \right) Q^{(p_2)}_{m_2} \right] \right\} \\
&\quad \left\{ I_{q_3} \otimes \left[\left(F_{p_3} \otimes I_{m_4} \right) Q^{(p_3)}_{m_3} \right] \right\} \\
&\quad \cdots \left\{ I_{q_{K-1}} \otimes \left[\left(F_{p_{K-1}} \otimes I_{m_K} \right) Q^{(p_{K-1})}_{m_{K-1}} \right] \right\} \\
&\quad \left\{ I_{q_K} \otimes \left[\left(F_{p_K} \otimes I_{m_{K+1}} \right) Q^{(p_K)}_{m_K} \right] \right\} \; P_N^{(p_1, p_2, \ldots, p_K)} \\
&= \prod_{j=1}^{K} \left\{ I_{q_j} \otimes \left[\left(F_{p_j} \otimes I_{m_{j+1}} \right) Q^{(p_j)}_{m_j} \right] \right\} \; P_N^{(p_1, p_2, \ldots, p_K)} \qquad (26)
\end{aligned}$$

for the time-decimation algorithm, and

$$F_N = P_N^{(p_K, p_{K-1}, \ldots, p_1)} \prod_{j=K}^{1} \left\{ I_{q_j} \otimes \left[Q^{(p_j)}_{m_j} \left(F_{p_j} \otimes I_{m_{j+1}} \right) \right] \right\} \qquad (27)$$

for the frequency-decimation algorithm, where $m_j \equiv N/(p_1 p_2 p_3 \ldots p_{j-1})$ and $q_j \equiv p_1 p_2 \cdots p_{j-1}$.

A modification of (26) and (27) is possible where the mixed-radix permutation is distributed through the calculation as a series of two-factor permutations. The result is

$$F_N = \prod_{j=1}^{K} A_j \left(P^{(q_j, p_j)}_{q_{j+1}} \otimes I_{m_{j+1}} \right) = \prod_{j=K}^{1} \left(P^{(p_j, q_j)}_{q_{j+1}} \otimes I_{m_{j+1}} \right) B_j, \qquad (28)$$

where

$$A_j = \left\{ I_{q_j} \otimes \left[\left(F_{p_j} \otimes I_{m_{j+1}} \right) Q^{(p_j)}_{m_j} \right] \right\}, \qquad (29)$$

$$B_j = \left\{ I_{q_j} \otimes \left[Q^{(p_j)}_{m_j} \left(F_{p_j} \otimes I_{m_{j+1}} \right) \right] \right\}. \qquad (30)$$

More details are given in Schatzman [9]. We note that the permutations in (28) can be accomplished without an explicit permutation, but by appropriate indexing in each block diagonal matrix-vector multiplication step

$$\sum_{l,m=0}^{N-1} (A_j)_{k,l} \left(P_{q_{j+1}}^{(q_j,p_j)} \otimes I_{m_{j+1}}\right)_{l,m} z_m = \sum_{m=0}^{N-1} (A_j)_{k,m'} z_m, \tag{31}$$

where the index mapping $m'(m)$ is computed according to

$$\begin{aligned} m' &= d_0 + d_2 m_{j+1} + d_1 m_{j+1} p_j, \\ m &= d_0 + d_1 m_{j+1} + d_2 m_{j+1} q_j, \end{aligned} \tag{32}$$

where the three-radix digits d_0, d_1, and d_2 satisfy $0 \le d_0 \le m_{j+1} - 1$, $0 \le d_1 \le q_j - 1$, and $0 \le d_2 \le p_j - 1$. This constitutes a block two radix reversed-digit mapping with a block size of m_{j+1}. These results are similar to those of Temperton [11].

Finally, the prime factor algorithm version of the FFT is nearly identical to the above, except that inputs and outputs are reordered, generally to effect the elimination of the twiddle factors (Burrus [1]; Schatzman [10]). Although the computations are reordered, and in some versions the underlying FFT blocks are modified by raising each element to a fixed integer power, the arguments above about stability of the stages of the algorithm still apply. The absence of twiddle factors could result in some improvement in accuracy; to this author's knowledge no results on this question have been published.

3.3. Stability of the FFT decomposition. If we examine (26)–(28) we see that the DFT is accomplished through a series of matrix-vector products. The permutation matrices have eigenvalues on the unit circle, with phases equal to multiples of $\frac{2\pi}{L}$, where L is the length of a permutation cycle associated with the matrix.

The matrices A_j and B_j are more complex. The matrix $F_{p_j} \otimes I_{m_{j+1}}$ has the same spectrum as F_{p_j}, namely eigenvalues $\pm\sqrt{p_j}$ and $\pm\imath\sqrt{p_j}$, but multiplied in multiplicity by the factor m_{j+1}. Likewise, the premultiplication $I_{q_j} \otimes C$ only multiplies the multiplicity of the eigenvalues of an arbitrary matrix C. Thus, the eigenvalues of A_j and B_j are modified only by the effect of postmultiplying and premultiplying $F_{p_j} \otimes I_{m_{j+1}}$ by $Q_{m_j}^{(p_j)}$, respectively. Because the matrices are unitary or scaled unitary, the eigenvalues of A_j and B_j all have magnitude $\sqrt{p_j}$; however, the phases of the eigenvalues are not the same as F_{p_j}. For example, the eigenvalues of $S_m^{(2)} = \left(F_2 \otimes I_{m/2}\right) Q_m^{(2)}$ are

$$\lambda_j = \frac{1}{2}\left(1 - w^j \pm \sqrt{(w^j - 1)^2 + 8w^j}\right), \qquad 0 \le j \le \frac{m}{2} - 1, \tag{33}$$

where $w = e^{-\imath 2\pi/m}$, which are irregularly distributed around the circle of radius $\sqrt{2}$ in the complex plane. From this elementary analysis, we see that no complications, such as increased sensitivity to numerical error due to increased condition number of the coefficient matrices, are to be expected from the FFT in any form. In fact, as we will see below, the FFT is very stable in most respects.

3.4. Numerical errors. Like the slow DFT, errors in the computation of the FFT consist of errors in the approximations of the sine/cosine phase factors and roundoff errors in multiplication and addition.

To test these predictions, slow DFTs were computed with single and double precision IEEE floating point. Figures 3 and 4 show a comparison of 128-bit floating point FFT results (taken to be truth) with

(3a) IEEE 32-bit FFT computations using the manufacturer's 32-bit sine/cosine functions and the FFTPAK recursion (see comments below),

(3b) IEEE 32-bit FFT computations modified to use accurate sine/cosine phase factors,

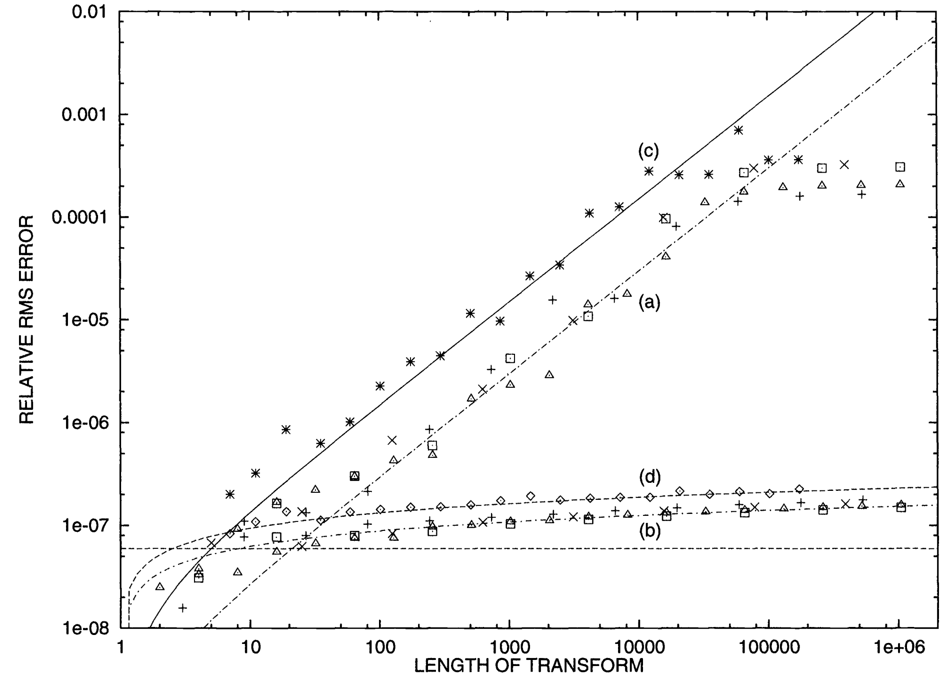


FIG. 3. *Relative RMS errors in the computed DFT for the* 32*-bit IEEE floating point FFT using* (a) *the IBM RS6000 library* 32*-bit sine and cosine functions and the NCAR FFTPAK table calculation code,* (b) *an accurate sine/cosine table,* (c) *the Schatzman algorithm* [9] *with inaccurate tables, and* (d) *the Schatzman algorithm* [9] *with accurate tables.* △, +, □, *and* × *denote lengths that are powers of* 2, 3, 4, *and* 5, *respectively;* $*$ *and* ◇ *denote the results for the Schatzman algorithm* [9] *with inaccurate and accurate tables, respectively. The interpolating error functions (smooth curves) are* $e_1(N) = \frac{1}{10}\epsilon_{32}(N-1)$ *for the conventional FFT with inaccurate tables,* $e_2(N) = 0.6\epsilon_{32}\sqrt{\log_2 N}$ *with accurate tables,* $e_3(N) = \frac{1}{2}\epsilon_{32}(N-1)$ *for the Schatzman algorithm* [9] *with inaccurate tables, and* $e_4(N) = 0.9\epsilon_{32}\sqrt{\log_2 N}$ *with accurate tables. The horizontal dashed line represents the constant* $\epsilon_{32} = 5.96 \cdot 10^{-8}$.

(4a) IEEE 64-bit FFT computations using the manufacturer's 32-bit sine/cosine functions and the FFTPAK recursion (see comments below),

(4b) IEEE 64-bit FFT computations modified to use accurate sine/cosine phase factors.

The NCAR FFTPAK (Version 2, February 1978) FFT code was used as the basis for these results. However, the package was modified in several ways:

(1) Minor modifications to the code were made to bring it into conformance with the ANSI Fortran 77 standard.

(2) Power-of-two transforms were evaluated using $p = 2$, not a mix of $p = 2$ and $p = 4$ as in the original code. This change was made so that the performance of the $p = 2$ code could be measured independently of the $p = 4$ code.

(3) For the accurate table tests, the 32- and 64-bit sine/cosine table computations were replaced with 64- or 128-bit precision computations, respectively.

Input data and analysis of the results were as above for the slow DFT.

Fits to the measured RMS errors were obtained for the following error functions:

(3a) $\frac{1}{10}\epsilon_{32}(N-1)$,

(3b) $0.6\epsilon_{32}\sqrt{\log_2 N}$,

(4a) $\frac{1}{20}\epsilon_{64}(N-1)$,

(4b) $0.6\epsilon_{64}\sqrt{\log_2 N}$.

Also shown as Figures 3c, 3d, 4c, and 4d are results for the new algorithm of Schatzman [9] for prime lengths. Conclusions from these results follow:

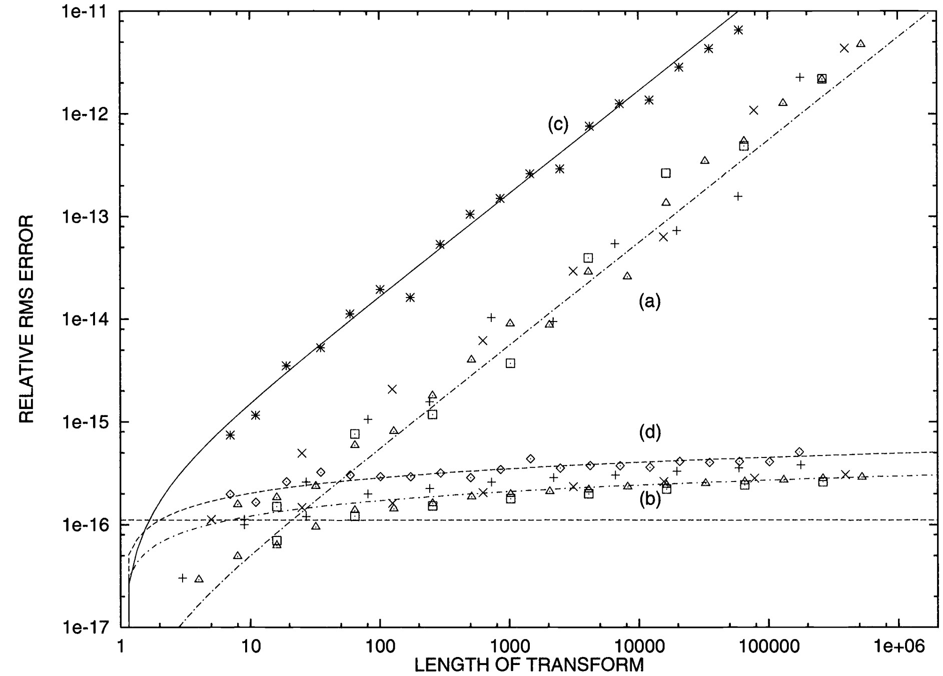


FIG. 4. *Relative RMS errors in the computed DFT for the* 64*-bit IEEE floating point FFT using* (a) *the IBM RS6000 library* 64*-bit sine and cosine functions and the NCAR FFTPAK table calculation code,* (b) *an accurate sine/cosine table,* (c) *the Schatzman algorithm* [9] *with inaccurate tables, and* (d) *the Schatzman algorithm* [9] *with accurate tables.* △, +, □, *and* × *denote lengths that are powers of* 2, 3, 4, *and* 5, *respectively;* $*$ *and* ◇ *denote the results for the Schatzman algorithm* [9] *with inaccurate and accurate tables, respectively. The interpolating error functions (smooth curves) are* $e_1(N) = \frac{1}{20}\epsilon_{64}(N-1)$ *for the conventional FFT with inaccurate tables,* $e_2(N) = 0.6\epsilon_{64}\sqrt{\log_2 N}$ *with accurate tables, and* $e_3(N) = 1.5\epsilon_{64}(N-1)$ *for the Schatzman algorithm* [9] *with inaccurate tables. The horizontal dashed line represents the constant* $\epsilon_{64} = 1.11 \cdot 10^{-16}$.

(1) Comparing Figures 1a and 3a, 1b and 2b, etc., we conclude that *FFTs typically produce approximate DFTs that are more accurate than the slow algorithm by a factor of at least* 10.

(2) Accuracy of the sine/cosine phase factors are crucial to the overall accuracy of the DFT. However, in the NCAR FFTPAK code, only part of the table is computed by explicit sine and cosine evaluations. The remainder of the table is generated recursively. Replacing the explicit sine/cosine evaluations with extremely accurate calculations (128-bit) made very little difference. The resulting plots are nearly indistinguishable from Figures 3a and 4a. *The recursive evaluation of entries of the sine/cosine table, as implemented in the NCAR FFTPAK code, is by far the largest single source of error.* The particular recursion used in the NCAR FFTPAK accentuates the errors of the initial sine/cosine approximations.

(3) When the sine/cosine table of the FFT is inaccurate (as in Figures 3a and 4a) the RMS DFT errors are roughly proportional to the length of the transform.

(4) When the sine/cosine table is accurate (Figures 3b and 4b) the RMS DFT errors grow slowly, apparently more slowly than the logarithm of the length of the transform.

(5) The computation of long DFTs with IEEE 32-bit floating point is a process about which one should be cautious. If accuracy of better than 0.1% is required (three digits) for transforms in the 10,000+ point range, 64-bit floating point or a very carefully implemented 32-bit FFT should be used.

(6) The Schatzman [9] algorithm is less accurate than the conventional FFT, but only modestly so.

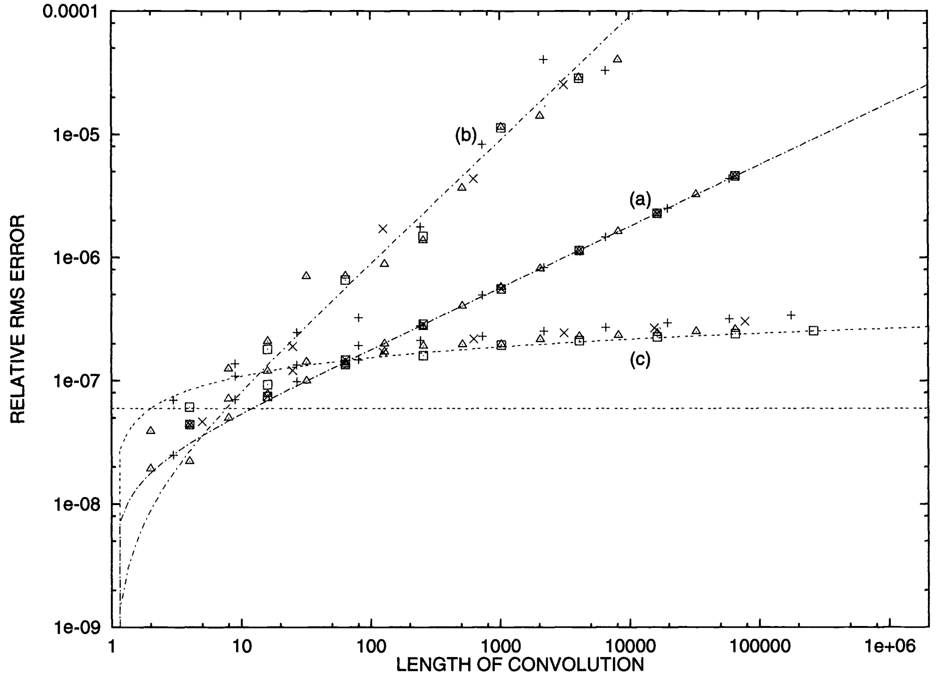


FIG. 5. *Relative RMS errors in the computed convolution for* (a) *the* 32-*bit IEEE floating point direct computation,* (b) *the FFT computation with inaccurate sine/cosine tables, and* (c) *with accurate tables.* △, +, □, *and* × *denote lengths that are powers of* 2, 3, 4, *and* 5, *respectively. The interpolating error functions (smooth curves) are* (a) $e_1(N) = 0.3\epsilon_{32}\sqrt{N{-}1}$, (b) $e_2(N) = 0.15\epsilon_{32}(N{-}1)$, *and* (c) $e_3(N) = \epsilon_{32}\sqrt{\log_2 N}$. *The horizontal dashed line represents the constant* $\epsilon_{32} = 5.96 \cdot 10^{-8}$.

4. Numerical errors for convolutions. We now study the computation of a circular convolution or cross-correlation via Fourier transform. Figures 5 and 6 show results for cross-correlation of white Gaussian complex sequences using direct computation and the comparable results using FFTs. We obtain fits to the data with the following functions:

(5a) Direct computation, 32-bit: $0.3\epsilon_{32}\sqrt{N{-}1}$;

(5b) FFT computation, 32-bit: $0.15\epsilon_{32}(N{-}1)$;

(5c) FFT computation, 32-bit with accurate sine/cosine table: $\epsilon_{32}\sqrt{\log_2 N}$;

(6a) Direct computation, 64-bit: $0.3\epsilon_{64}\sqrt{N{-}1}$;

(6b) FFT computation, 64-bit: $0.2\epsilon_{64}(N{-}1)$;

(6c) FFT computation, 64-bit with accurate sine/cosine table: $\epsilon_{64}\sqrt{\log_2 N}$.

These errors are very similar to the errors for the slow and fast DFT themselves, as reported above.

We recall the claim of Gottlieb and Orszag [5]: "Transform methods normally give no appreciable amplification of roundoff errors. In fact, the evaluation of convolution-like sums using FFTs often gives results with much smaller roundoff error than would be obtained if the convolution sums were evaluated directly." We have quantitatively confirmed these observations; the FFT method gives more accurate results for vector lengths of approximately 100 and greater with IEEE 32- and 64-bit floating point. However, this conclusion depends on having accurate FFT software. For example, the stock NCAR FFTPAK software gives less accurate results for the convolution than direct computation by arithmetic of the same precision. Asymptotically for large vector lengths, the RMS error is proportional to the square root of the logarithm of the vector length for the (accurate) FFT method and proportional to

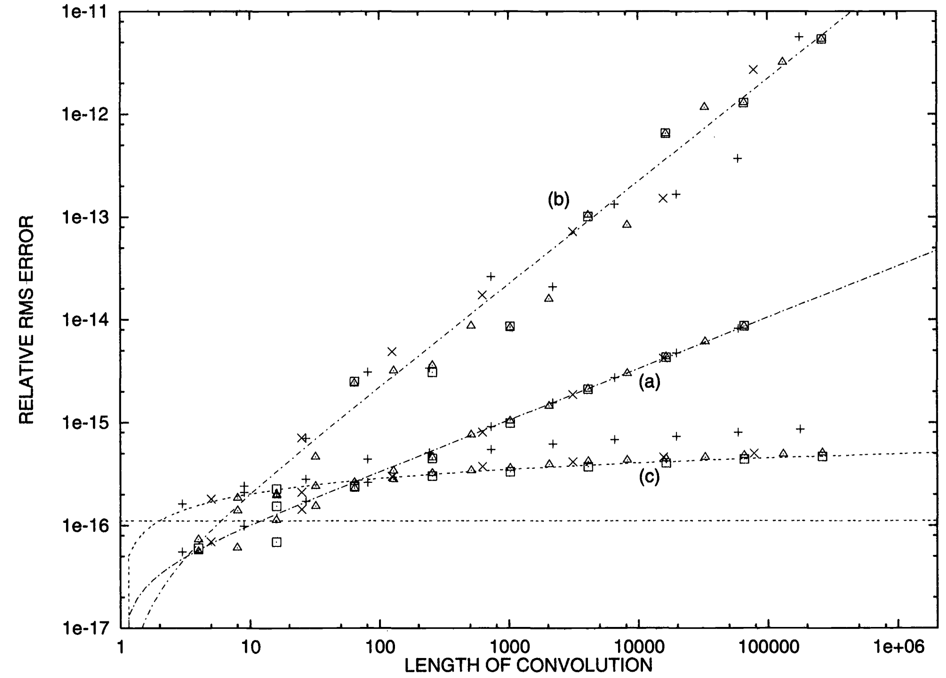


FIG. 6. *Relative RMS errors in the computed convolution for* (a) *the* 64-*bit IEEE floating point direct computation,* (b) *the FFT computation with inaccurate sine/cosine tables, and* (c) *with accurate tables. The interpolating error functions (smooth curves) are* (a) $e_1(N) = 0.3\epsilon_{64}\sqrt{N-1}$, (b) $e_2(N) = 0.2\epsilon_{64}(N-1)$, *and* (c) $e_3(N) = \epsilon_{64}\sqrt{\log_2 N}$. *The horizontal dashed line represents the constant* $\epsilon_{64} = 1.11 \cdot 10^{-16}$.

the square root of the vector length for the direct method. The explanation is that the FFT involves $\log(N)$ additions of uncorrelated errors for each component of the result, whereas the direct computation involves N–1 additions of uncorrelated errors.

5. Error in the twiddle factors. As observed above, the twiddle factors can be the dominant source of error. In some cases the library software for the sine/cosine is inaccurate. For example, on the Cray Y/MP, it appears that the decision was made to provide speed instead of accuracy. Accuracy of standard library routines should be tested; high-accuracy routines should be substituted if the library routines are inaccurate, keeping in mind that speed is generally irrelevant for this application (initiation of the FFT).

How to design algorithms that produce high-accuracy (one-half machine epsilon) sine/consine functions given floating point arithmetic of only machine epsilon accuracy is beyond the scope of this paper. I have found that it is difficult to obtain highly accurate sine and cosine functions using standard Taylor series or continued fraction expansions with floating hardware that does not provide double precision intermediary products. For example, the IBM RS/6000 processors provide high-accuracy (rounded) products whereas popular Intel and MIPS processors do not. After considerable experimentation, I offer the following conclusions and advice:

(1) Empirically, the error in accurate rounded 32- and 64-bit IEEE twiddle factor values as required by the FFT is essentially uniformly distributed on $[-\epsilon/2, \epsilon/2]$ for all n between 16 and 131,072. In other words, there is nothing peculiar about the twiddle factors that would upset this standard assumption. Therefore, the standard deviation of error in optimal twiddle factors is $\epsilon/\sqrt{12}$.

(2) The error obtained by direct library call to sine/cosine functions with the best libraries I tested is somewhat greater than the optimum—nearly 1.3–1.5 times the optimum. With other libraries the error can be much greater. Direct implementation of Taylor series leads to errors (with full-rounding floating point) of nearly 0.7ϵ. Kahan's summation method (Higham [6]) added to the Taylor series computation reduces the error to the theoretically optimal $\epsilon/\sqrt{12}$ or below.

(3) Errors resulting from the use of the recursion (34) are typically all of one sign (more about this below). The point is that the errors in the sine/consine table when computed this way are definitely not random (that is, not uniformly distributed over $[-\epsilon/2, \epsilon/2]$). A consequence is that overall FFT errors can be much larger than anticipated.

(4) If higher-precision arithmetic is available, by all means I recommend using that higher precision to compute the twiddle factors, which should then be rounded to the desired precision. Of course, arbitrary precision software is available (public domain) but the slowness may be objectionable.

(5) The sine/cosine tables could be pretabulated to high accuracy and recorded in binary form. Once this is done for a given floating point format, the code is portable to machines using that format. If implemented intelligently, the amount of disk space required for a complete tabulation for modestly large N is quite reasonable.

Some standard FFT packages use the following recursion for the sine and cosine functions:

$$(34)\qquad \begin{cases} C_n = \cos\theta\ C_{n-1} - \sin\theta\ S_{n-1}, \\ S_n = \cos\theta\ S_{n-1} + \sin\theta\ C_{n-1} \end{cases}$$

for $n = 1, 2, \ldots, C_0 = \cos\theta_0$, and $S_0 = \sin\theta_0$. Replacing $\cos\theta$ by the approximate value c and $\sin\theta$ by the approximate value s, we obtain

$$(35)\qquad \begin{pmatrix} C \\ S \end{pmatrix}_n = \begin{pmatrix} c & -s \\ s & c \end{pmatrix} \begin{pmatrix} C \\ S \end{pmatrix}_{n-1}.$$

The eigenvalues of the coefficient matrix are $\lambda = c \pm \imath s$, and

$$(36)\qquad \lambda^N = (c^2 + s^2)^{\frac{N}{2}} e^{\imath N \tan^{-1} \frac{s}{c}}.$$

Applying asymptotic analysis to (36), we see that errors in C_n and S_n grow at first linearly and later exponentially with the length of the recursion. This is highly undesirable; what is needed is a self-correcting recursion in which errors do not grow. While this analysis ignores roundoff error, actual calculations verify this exponential growth.

Chu [3] analyses the recursion (34), including the effects of roundoff, and concludes that errors grow linearly. This analysis is erroneous, resulting from ignoring errors in sines in the analysis of the cosines and vice versa. Oliver [8] gives an excellent review of recursions known at that time, but does not include (34). Chu [3] describes (34) and many other algorithms; some of the results of this paper are in error, however. Also, (34) can easily be improved as follows:

$$(37)\qquad \begin{cases} A_n = \cos\theta\ C_{n-1} - \sin\theta\ S_{n-1} + \sin\theta\ \alpha, \\ B_n = \cos\theta\ S_{n-1} + \sin\theta\ C_{n-1}, \\ C_n = \frac{A_n}{\sqrt{A_n^2 + B_n^2}}, \\ S_n = \frac{B_n}{\sqrt{A_n^2 + B_n^2}}, \end{cases}$$

where α depends on N and the type of floating point hardware. Thus, we correct both the phase and amplitude of the sine/cosine pair at each step. This improved method is based on the empirical observations that the errors in (34) are patterned. However, the results of the best recursions are still significantly worse than accurate direct calculations.

My recommendation is that the twiddle factors be computed to high accuracy, either with high-precision arithmetic or Kahan's summation method (if Kahan's method works effectively on the given hardware). Truncated Taylor's series work very well for this purpose. If speed in initialization is required, depending on the machine architecture, either twiddle factors should be precomputed and stored on disk or faster converging approximations such as rational or continued fraction approximations should be used. Finally, if a recursive computation must be used for some reason, (34) should never be used but a more stable recursion (Oliver [8] and Chu [3]) should be employed.

6. Two and higher dimensions. I have not undertaken a thorough examination of the two- and higher-dimensional problem to verify that the one-dimensional results extend in a natural way. However, cursory examination of the problem suggests that higher-dimensional FFTs are even more remarkably stable, at least if the size of the dimensions is large and the number of dimensions is small.

For example, 1024-by-1024 transforms were executed with white random data using IEEE 64-bit floating point arithmetic. Using the manufacturer's sine/cosine functions and the usual recursions, RMS relative errors of $200\epsilon_{64}$ were observed. Using accurate sine/cosine tables, RMS relative errors of less than $3\epsilon_{64}$ were observed. These two-dimensional transforms were done in the usual way through repeated application of one-dimensional transforms along the rows and columns. Interestingly, the differences between the rows first/columns last and columns first/rows last results were observed to be approximately $4\epsilon_{64}$ regardless of the accuracy of the sine/cosine table. Finally, averaging the row/column and column/row results for the accurate sine/cosine table gave RMS relative errors of approximately $2\epsilon_{64}$. This is very little larger than the error for a one-dimensional transform of length 1024 $(1.9\epsilon_{64})$. It is remarkable that the transforms along the second dimension do not increase the error more than this. To lose only one bit of accuracy after thousands of floating point operations dramatizes the great stability of the FFT, when implemented carefully!

7. Conclusions. When an accurate sine/cosine table is used, the FFT is remarkably stable. However, the accuracy is greatly impaired by inaccurate sine/cosine tables. Inaccuracy in the sine/cosine table is common and results from inaccurate underlying software library functions and ill-advised recursions for table construction.

Acknowledgments. The anonymous reviewers were very helpful in drawing the author's attention to relevant literature.

REFERENCES

[1] C. S. Burrus, *Index mappings for multidimensional formulation of the DFT convolution*, IEEE Trans. Acoustics, Speech, Signal Processing, 25 (1977), pp. 239–42.

[2] D. Calvetti, *A stochastic roundoff error analysis for the fast Fourier transform*, Math. Comp., 56 (1991), pp. 755–774.

[3] C. Y. Chu, *The Fast Fourier Transform on Hypercube Parallel Computers*, Technical Report 87-882, Dept. of Computer Science, Cornell University, Ithaca, NY, 1987.

[4] W. M. Gentleman and G. Sande, *Fast Fourier transforms–For fun and profit*, in 1966 Fall Joint Computer Conference, AFIPS Conf. Proceedings #29, Spartan, Washington, D.C., pp. 563–578.

[5] D. Gottlieb and S. A. Orszag, *Numerical Analysis of Spectral Analysis: Theory and Applications*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1977.

[6] N. J. Higham, *The accuracy of floating-point summation*, SIAM J. Sci. Comput., 14 (1993), pp. 783–799.

[7] T. Kaneko and B. Liu, *Accumulation of round-off error in fast Fourier transforms*, J. Assoc. Comput. Mach., 17 (1970), pp. 637–654.

[8] J. Oliver, *Stable methods for evaluating the points* $\cos(i\pi/n)$, J. Inst. Math. Applic., 16 (1975), pp. 247–257.

[9] J. Schatzman, *Fast Fourier transform algorithms and complexity of the discrete Fourier transform*, Math. Comp., submitted.

[10] ———, *Index mappings for the fast Fourier transform*, IEEE Trans. Signal Processing, 44 (1996).

[11] C. Temperton, *Self-sorting mixed-radix fast Fourier transforms*, J. Comput. Phys., 52 (1983), pp. 1–23.

[12] C. Van Loan, *Computational Frameworks for the Fast Fourier Transform*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.

SIAM J. SCI. COMPUT.
Vol. 17, No. 5, pp. 1167–1174, September 1996

009

COMPUTING THE EXTREMAL POSITIVE DEFINITE SOLUTIONS OF A MATRIX EQUATION*

XINGZHI ZHAN†

Abstract. An efficient and numerically stable implementation of a known algorithm is suggested for finding the extremal positive definite solutions of the matrix equation $X+A^*X^{-1}A=I$, if such solutions exist. The convergence rate is analyzed. A new algorithm that avoids matrix inversion is presented. Numerical examples are given to illustrate the effectiveness of the algorithms.

Key words. matrix equation, positive definite solution, iteration

AMS subject classifications. 65F10, 65F30

1. Introduction.

The problem of positive definite solutions of the matrix equation $X+A^*X^{-1}A=Q$, with Q positive definite, arises in many applications. These areas include control theory [7, 10, 16, 17], ladder networks [1, 4, 18], dynamic programming [14], stochastic filtering [2, 6], and statistics [6, 13]. Recently this equation has been studied by some authors [3, 7, 8, 19] and they mainly concentrated on the theoretical properties such as necessary and sufficient conditions for the existence of a positive definite solution. Obviously the equation can be reduced to

$$X+A^*X^{-1}A=I, \tag{1.1}$$

where I is the identity matrix. Note that the equation may have non-Hermitian or indefinite solutions. For example, if $A=I_2$, then

$$X=\begin{pmatrix} 1 & -\alpha \\ \alpha^{-1} & 0 \end{pmatrix},$$

with any real number $\alpha \neq 0$, is always a solution. We will not consider this kind of solution in our applications. Throughout this paper, a solution means a Hermitian positive definite one. Since the equation is highly nonlinear, designing an efficient algorithm does not seem easy. We will take one step in this direction.

Let matrices B and C be Hermitian. If $B-C$ is positive definite (semidefinite), we write $B>C(B\geq C)$. It was proved in [8] that if (1.1) has a solution, then it has a maximal solution X_L and a minimal solution X_S in the sense that for any solution X, $X_S\leq X\leq X_L$. If A is invertible, then it is easy to verify (see [8, Thm. 3.3]) that X_S is the minimal solution of (1.1) if and only if $Y_L=I-X_S$ is the maximal solution of $Y+AY^{-1}A^*=I$. Thus any algorithm for computing X_L is also one for computing X_S, and in the paper we only discuss how to compute X_L.

If A is a normal matrix, we have formulae for both X_L and X_S, which are conjectured in [7] and proved in [19]. From the computational point of view, in the case where A is singular the problem of obtaining the minimal solution X_S is not solved here and remains a topic of future research. If A is singular, theoretically we can always reduce (1.1) to the nonsingular case of the same form in lower dimensions [8, §3], but this is not practical. Perhaps there are some direct methods for computing X_S without this reduction.

In §2 we analyze the convergence rate of a known algorithm and the implementation of it. A new algorithm that avoids matrix inversion is presented in §3. Numerical examples are given in §4.

*Received by the editors November 9, 1994; accepted for publication (in revised form) April 28, 1995.

†Institute of Mathematics, Fudan University, Shanghai 200433, P.R. China. Current address: Institute of Mathematics, Peking University, Beijing 100871, P.R. China. This research was supported by the National Science Foundation of China.

2. Implementation of a known algorithm and analysis of its convergence rate. To solve (1.1) we can use the following natural iterative procedure.

ALGORITHM 2.1.

$$X_0 = I, \qquad X_{n+1} = I - A^* X_n^{-1} A. \tag{2.1}$$

It was proved in [7] that (1.1) has a solution if and only if $X_n > AA^*$ for all n. If (1.1) has a solution, then X_n is a monotone decreasing sequence and converges to the maximal solution X_L [8]. It is said in [8, §4] that whether both solutions X_L and X_S are obtained from this algorithm in a numerically reliable way remains an open question. We will show that it is indeed so with the implementation here. What we should consider about the implementation of this algorithm is how to compute $A^* X_n^{-1} A$. In most cases A is real and now we make this assumption. Suppose that (1.1) has a solution. The computations may be done as follows.

Step 1. Computing the Cholesky factorization of X_n, $X_n = LL^T$.

Very effective methods for this factorization are available and they are numerically stable. See [9, §4.2].

Step 2. Solving the triangular systems

$$Lb_i = a_i, \quad i = 1, 2, \ldots, m,$$

where a_i are the columns of A, $A = (a_1, \ldots, a_m)$. Set $B = (b_1, \ldots, b_m)$.

We may use any algorithms described in [9, §3.1] and it is pointed out that the accuracy of a computed solution to a triangular system is often surprisingly good. If the order of the matrices is large enough, we use the block forward substitution algorithm [9, §3.1.4] to $LB = A$, which is rich in matrix multiplication.

Step 3. Computing $B^T B$ and forming $X_{n+1} = I - B^T B$.

Note that using the symmetry we need only to compute half of the entries of $B^T B$, say, the upper triangular part. As to the storage, we should also use the symmetry.

If $A \in \mathbb{R}^{m \times m}$, then the algorithm costs $\frac{m^3}{3} + m^3 + m^3 = \frac{7}{3}m^3$ flops per iteration. Now we examine the convergence rate. In what follows we denote by $\|\cdot\|$ the spectral norm. Since X_n is monotone decreasing and converges to X_L, $0 \leq X_{n+1} - X_L \leq X_n - X_L$, and thus $\|X_{n+1} - X_L\| \leq \|X_n - X_L\|$.

THEOREM 2.2. *Suppose that* (1.1) *has a solution. If* $\|X_L^{-1} A\| < 1$, *then* X_n *converges to* X_L *with at least the linear convergence rate.*

Proof. The convergence of X_n has been proved in [8]. From (2.1) and

$$X_L = I - A^* X_L^{-1} A,$$

we have

$$\begin{aligned} X_{n+1} - X_L &= A^*(X_L^{-1} - X_n^{-1})A \\ &= A^* X_n^{-1}(X_n - X_L) X_L^{-1} A. \end{aligned}$$

Thus

$$\begin{aligned} \|X_{n+1} - X_L\| &\leq \|A^* X_n^{-1}\| \cdot \|X_L^{-1} A\| \cdot \|X_n - X_L\| \\ &= \|X_n^{-1} A\| \cdot \|X_L^{-1} A\| \cdot \|X_n - X_L\|. \end{aligned}$$

Choose a real number θ satisfying $\|X_L^{-1} A\| < \theta < 1$. Since $X_n \to X_L$, there exists a k such that for any $n \geq k$, $\|X_n^{-1} A\| \leq \theta$. Hence

$$\|X_{n+1} - X_L\| \leq \theta^2 \|X_n - X_L\|. \qquad \square$$

COROLLARY 2.3. *Let X_n be the sequence in Algorithm* 2.1. *If* $\|A\| < 1/2$, *then* X_n *converges to* X_L *with at least the linear convergence rate.*

Proof. It is proved in [7, Thm. 13] that if $\|A\| < 1/2$, then (1.1) has a solution and thus the maximal solution X_L exists. By Theorem 2.2 we need only to show $\|X_L^{-1}A\| < 1$. Choose a number δ satisfying $\|A\| < \delta < 1/2$. We prove $\|X_n^{-1}A\| \leq \delta$ for all n by induction. First, $\|X_0^{-1}A\| = \|A\| < \delta$. Assume that $\|X_k^{-1}A\| \leq \delta$. From (2.1) we have

$$X_{k+1}^{-1}A = (I - A^*X_k^{-1}A)^{-1}A.$$

Using the expansion

$$(I - A^*X_k^{-1}A)^{-1} = \sum_{p=0}^{\infty}(A^*X_k^{-1}A)^p$$

and the assumption, we get

$$\begin{aligned}\|(I - A^*X_k^{-1}A)^{-1}\| &\leq 1/(1 - \|A^*X_k^{-1}A\|)\\ &\leq 1/(1 - \|A^*\| \cdot \|X_k^{-1}A\|)\\ &\leq 1/(1 - \|A\|\delta).\end{aligned}$$

Thus

$$\|X_{k+1}^{-1}A\| \leq \|A\|/(1 - \|A\|\delta) \leq \delta.$$

The last inequality can be easily verified. Notice that $(1-\sqrt{1 - 4\|A\|^2})/2 \leq \|A\| < \delta < 1/2$. Therefore $\|X_n^{-1}A\| \leq \delta$ for all n. Letting $n \to \infty$ yields $\|X_L^{-1}A\| \leq \delta < 1$. □

Remark 2.4. From [8, Thm. 3.4] we have that X_L is the unique solution for which $X + \lambda A$ is invertible for all $|\lambda| < 1$. Now, $X_L + \lambda A = X_L(I + \lambda X_L^{-1}A)$. So $I + \lambda X_L^{-1}A$ is invertible for all $|\lambda| < 1$. From this it follows that the spectral radius $\rho(X_L^{-1}A) \leq 1$. So, in view of Theorem 2.2, whether this property also implies that $\|X_L^{-1}A\| \leq 1$ whenever (1.1) has a solution remains an open question. Note that using [19, Thm. 2.1] this last question can be reformulated as follows: Given square matrices W and Z satisfying $W^*W + Z^*Z = I$, with the additional properties that W is invertible and $\rho(W^{-1}Z) \leq 1$, does this imply that $\|W^{-1}Z\| \leq 1$?

Remark 2.5. Linear convergence means possibly very slow convergence. It would be of great interest to get a quadratically convergent iteration.

3. A new algorithm. Golub and Van Loan [9, p. 9] said that the design of matrix algorithms that are rich in high-level linear algebra operations is an area of intensive research. To pursue a computational simplicity we develop an algorithm for (1.1) that is of level 3. See [9, p. 9] for the notion of "level." It involves only matrix–matrix multiplication. The basic idea is to replace computing the inverse X_n^{-1} in Algorithm 2.1 by another simultaneous iteration. This is reasonable since Algorithm 2.1 is itself an iterative procedure. The Schulz iteration [15] for A^{-1} has been used to compute the Moore–Penrose generalized inverse [5]. Let $0 < \alpha < 2\|A\|^{-2}$,

$$\begin{cases} X_0 = \alpha A^*,\\ X_{k+1} = X_k(2I - AX_k).\end{cases}$$

Then $\lim_{k\to\infty} X_k = A^+$. It is easy to check that X_k is quadratically convergent. Recently this iteration was also used in [12]. Incorporating the Schulz iteration into Algorithm 2.1 yields our next algorithm.

ALGORITHM 3.1.

$$X_0 = Y_0 = I,$$

$$\begin{cases} X_{n+1} = I - A^*Y_nA, \\ Y_{n+1} = Y_n(2I - X_nY_n). \end{cases} \tag{3.1}$$

If (1.1) has a solution, then X_n is monotone decreasing and converges to X_L. In the case where A is nonsingular, this procedure can also be used to decide whether (1.1) has a solution.

For real numbers p and c with $p > 0$, we have $c^2p + p^{-1} \geq 2c$. The matrix analogue of this fact is also true. The following result will play a crucial role in the analysis of the convergence.

LEMMA 3.2. *Let C and P be Hermitian matrices of the same order and let $P > 0$. Then*

$$CPC + P^{-1} \geq 2C. \tag{3.2}$$

Proof. Let $P = Q^*Q$ for some nonsingular Q. Rewrite (3.2) as

$$CQ^*QC + Q^{-1}Q^{-*} \geq 2C$$

or, equivalently, as

$$QCQ^*QCQ^* + I \geq 2QCQ^*.$$

Since C is Hermitian, the above inequality is equivalent to

$$(QCQ^* - I)^*(QCQ^* - I) \geq 0,$$

which is obviously true. □

Now we investigate the properties of Algorithm 3.1. The fact that $A \geq B > 0$ implies that $0 < A^{-1} \leq B^{-1}$ will be used.

THEOREM 3.3. *Let X_n be the iterates in Algorithm* 3.1. *If* (1.1) *has a positive definite solution, then the sequence X_n is monotone decreasing and converges to the maximal solution X_L. Moreover, if the matrix A is nonsingular and all $X_n > 0$, then* (1.1) *has a positive definite solution.*

Proof. Suppose that (1.1) has a solution. We first prove by induction that each $X_n \geq X_L$ and the sequence of Y_n is monotone increasing. From

$$X_L = I - A^*X_L^{-1}A, \tag{3.3}$$

we have $X_0 = I \geq X_L$, $X_1 = X_2 = I - A^*A \geq X_L$, and $Y_0 = Y_1 = I \leq I + A^*A = Y_2$.

Suppose $X_k \geq X_L$ and $Y_k \geq Y_{k-1}$ for all $k \leq n, n \geq 2$. Note that every X_i and Y_i is Hermitian and $Y_n \geq I$. Using Lemma 3.2 with $C = Y_{n-1}$ and $P = X_n$, we have

$$X_n^{-1} \geq 2Y_{n-1} - Y_{n-1}X_nY_{n-1}. \tag{3.4}$$

From (3.1) we get

$$X_{n-1} - X_n = A^*(Y_{n-1} - Y_{n-2})A. \tag{3.5}$$

But $Y_{n-1} \geq Y_{n-2}$, so $X_{n-1} \geq X_n$ and, therefore,

$$2Y_{n-1} - Y_{n-1}X_nY_{n-1} \geq 2Y_{n-1} - Y_{n-1}X_{n-1}Y_{n-1} = Y_n. \tag{3.6}$$

Combining (3.4) with (3.6) yields

$$X_n^{-1} \geq Y_n \quad \text{or} \quad Y_n^{-1} \geq X_n. \tag{3.7}$$

The assumption $X_n \geq X_L$ implies $X_n^{-1} \leq X_L^{-1}$. From (3.7) we get

$$Y_n \leq X_L^{-1}. \tag{3.8}$$

Thus $X_{n+1} = I - A^*Y_nA \geq I - A^*X_L^{-1}A = X_L$. Rewrite the second formula of (3.1) as

$$Y_{n+1} - Y_n = Y_n(Y_n^{-1} - X_n)Y_n.$$

From (3.7) and the above expression we obtain $Y_{n+1} \geq Y_n$. This completes the induction. As Y_n is monotone increasing, by (3.5) we know that X_n is monotone decreasing. Since Y_n is monotone increasing and bounded from above by X_L^{-1} because of (3.8), $\lim_{n\to\infty} Y_n = Y$ exists and thus from (3.1) $\lim_{n\to\infty} X_n = X$ exists. Taking limits in (3.1) gives $Y = X^{-1}$ and $X = I - A^*X^{-1}A$. As each $X_n \geq X_L$, $X = X_L$.

Now suppose that A is nonsingular and all $X_n > 0$. Then the above proof of the monotonicity of Y_n remains valid. It follows that X_n is monotone decreasing and bounded from below by 0, so $\lim_{n\to\infty} X_n = X \geq 0$ exists. Since A is nonsingular, (3.1) implies that $Y_n = A^{-*}(I - X_{n+1})A^{-1}$. Therefore $\lim_{n\to\infty} Y_n = Y$ also exists. As $Y_0 = I$ and Y_n is monotone increasing, $Y \geq I$. Taking limits in (3.1) yields

$$X = I - A^*YA,$$
$$Y = YXY.$$

Since $Y \geq I$, $X = Y^{-1} > 0$, and thus $X = I - A^*X^{-1}A$. So (1.1) has a solution. □

The implementation of this algorithm is very simple. The symmetry should be exploited. If the order of A is m, then $6m^3$ flops are needed per iteration. The number of flops is larger than that of Algorithm 2.1 per iteration. By the monotonicity of X_n, $0 \leq X_{n+1} - X_L \leq X_n - X_L$, and thus $\|X_{n+1} - X_L\| \leq \|X_n - X_L\|$. For computational simplicity, we use the row sum norm $\|\cdot\|_\infty$. Note that from (3.1) $I - X_nY_n = Y_n^{-1}(Y_{n+1} - Y_n) \to 0$, as $n \to \infty$. Given a small number $\epsilon > 0$, one stopping criterion may be

$$\|I - X_nY_n\|_\infty < \epsilon. \tag{3.9}$$

Checking this criterion adds very little amount of work, since in view of (3.1) the product X_nY_n must be computed in the iteration itself. The effect of the criterion can be seen from the following result.

THEOREM 3.4. *Suppose* (1.1) *has a solution. If after n iterative steps of Algorithm* 3.1 *we have* $\|I - X_{n-1}Y_{n-1}\| < \epsilon$ *and* $\|I - X_nY_n\| < \epsilon$, *then*

$$\|X_n + A^*X_n^{-1}A - I\| \leq 2\|A\|^2\|X_L^{-1}\|\epsilon.$$

Proof.

$$\begin{aligned} X_n + A^*X_n^{-1}A - I &= X_n - X_{n+1} + A^*(X_n^{-1} - Y_n)A \\ &= A^*(Y_n - Y_{n-1})A + A^*(X_n^{-1} - Y_n)A \\ &= A^*[Y_{n-1}(I - X_{n-1}Y_{n-1}) + X_n^{-1}(I - X_nY_n)]A. \end{aligned}$$

Since $0 < Y_{n-1} \leq X_{n-1}^{-1} \leq X_L^{-1}$ and $0 < X_n^{-1} \leq X_L^{-1}$, we have

$$\|Y_{n-1}\| \leq \|X_L^{-1}\|, \quad \|X_n^{-1}\| \leq \|X_L^{-1}\|.$$

Thus

$$\begin{aligned}\|X_n + A^*X_n^{-1}A - I\| &\le \|A\|^2(\|Y_{n-1}\|\|I - X_{n-1}Y_{n-1}\| + \|X_n^{-1}\|\|I - X_nY_n\|)\\ &\le 2\|A\|^2\|X_L^{-1}\|\epsilon. \qquad \square\end{aligned}$$

Note that if (1.1) has a solution, then $\|A\| < 1$. If $A \in \mathbb{C}^{m\times m}$ and we have $\|I - X_{n-1}Y_{n-1}\|_\infty < \epsilon$ and $\|I - X_nY_n\|_\infty < \epsilon$, then $\|X_n + A^*X_n^{-1}A - I\| \le 2\|A\|^2\|X_L^{-1}\|\sqrt{m}\epsilon$. Usually this is an overestimation since we have used the relation $\|G\| \le \sqrt{m}\|G\|_\infty$ for $G \in \mathbb{C}^{m\times m}$.

According to Theorem 3.3, if we do not know whether (1.1) has a solution and A is nonsingular, then Algorithm 3.1 can also be used, and after some number of iterative steps, we begin to check the positive definiteness of X_n. If $X_n > 0$ we go up to some step n large enough and we claim numerically that (1.1) has a solution. Otherwise we stop and affirm that (1.1) has no positive definite solution. The Cholesky factorization is a good approach to test the positive definiteness. See the discussion in [11, §5].

Analysis of the convergence rate of Algorithm 3.1 seems hard. In particular, the author does not know whether this algorithm also has a linear convergence rate and he conjectures that this is the case.

4. Numerical examples. We carry out the following numerical experiments on an IBM-PC 286 computer with single precision. The machine precision is approximately 5.96×10^{-8}.

Example 1.

$$A = \begin{pmatrix} .1000000 & -.1500000 & -.2598076 \\ .1500000 & .2125000 & -.0649519 \\ .2598076 & -.0649519 & .1375000 \end{pmatrix}.$$

A is a normal matrix chosen at random so that (1.1) has a solution. In the case where A is normal and (1.1) has a solution (iff $\|A\| \le 1/2$, see [7]), it is conjectured in [7] and proved in [19, Thm. 4.1] that $X_L = \frac{1}{2}[I + (I - 4A^*A)^{1/2}]$. According to this formula the accurate maximal solution is

$$X_L = \begin{pmatrix} .88729835 & .00000000 & .00000000 \\ & .92158407 & -.01979489 \\ \mathit{symm.} & & .89872694 \end{pmatrix}.$$

After seven iterations of Algorithm 2.1 we get

$$\widetilde{X}_L = \begin{pmatrix} .88729840 & .00000000 & .00000001 \\ & .92158410 & -.01979488 \\ \mathit{symm.} & & .89872700 \end{pmatrix}.$$

Note that each entry of $\widetilde{X}_L$ has at least six correct digits. Ten iterations and 13 iterations of this algorithm give the same result:

$$\widetilde{X}_L = \begin{pmatrix} .88729830 & .00000000 & .00000001 \\ & .92158410 & -.01979491 \\ \mathit{symm.} & & .89872690 \end{pmatrix}.$$

Each entry of $\widetilde{X}_L$ has at least seven correct digits. After seven iterations of Algorithm 3.1, we get

$$\widetilde{X}_L = \begin{pmatrix} .88730160 & .00000000 & .00000000 \\ & .92158500 & -.01979354 \\ \mathit{symm.} & & .89872940 \end{pmatrix}.$$

The entry $\widetilde{X}_L(1,1)$ has four correct digits and other entries have at least six correct digits. Ten iterations of Algorithm 3.1 give

$$\widetilde{X}_L = \begin{pmatrix} .88729850 & .00000000 & .00000000 \\ & .92158410 & -.01979486 \\ \mathit{symm.} & & .89872700 \end{pmatrix}.$$

The entry $\widetilde{X}_L(3,3)$ has six correct digits and other entries have at least seven correct digits.

Example 2.

$$A = \frac{1}{40}\begin{pmatrix} 2 & -1 & 3 & 4 \\ 7 & 6 & -5 & 9 \\ 4 & 8 & 10 & 6 \\ -3 & 5 & 2 & 8 \end{pmatrix}.$$

A is nonnormal. We use the criterion $\|X_n - X_{n+1}\|_\infty = \|X_n + A^*X_n^{-1}A - I\|_\infty < tol$ for Algorithm 2.1. If $tol = 10^{-4}$, eight iterations are required. If $tol = 10^{-6}$, 11 iterations are required. If $tol = 10^{-8}$, 13 iterations are required. Use the criterion $\|I - X_nY_n\|_\infty < tol$ for Algorithm 3.1. If $tol = 10^{-4}$, 12 iterations are required. If $tol = 10^{-6}$, 18 iterations are required. Therefore we affirm numerically that for this A, (1.1) has a solution. One of the computed results is

$$\widetilde{X}_L = \begin{pmatrix} .94687310 & -.04486766 & -.00670385 & -.05718690 \\ & .89817360 & -.04311119 & -.11904720 \\ & & .90854990 & -.03544480 \\ \mathit{symm.} & & & .82728140 \end{pmatrix}.$$

Example 3. Modifying a scalar multiple of the matrix in Example 2 we get

$$A = \frac{1}{32}\begin{pmatrix} 2 & -1 & 3 & 4 \\ 7 & 6 & -5 & 9 \\ 4 & 8 & 10 & 6 \\ -3 & 5 & 2 & 8 \end{pmatrix}.$$

When using Algorithm 2.1, at the sixth implementation of the iteration a square root of negative argument appears, which occurs in the Cholesky factorization. This means that X_5 is not positive definite. Therefore we conclude that for this A, (1.1) has no positive definite solution.

The above examples show that both Algorithms 2.1 and 3.1 are numerically reliable methods either for judging the existence of positive definite solutions of (1.1) or for computing the extremal solutions X_L and X_S. Algorithm 2.1 converges slightly faster than Algorithm 3.1 in the sequential computations.

Acknowledgments. The author wishes to thank Erxiong Jiang for his continuous advice and encouragement and two anonymous referees for their helpful comments which improved the paper.

REFERENCES

[1] W. N. Anderson et al., *Ladder networks, fixed points, and the geometric mean*, Circuits Systems Signal Process., 3 (1983), pp. 259–268.

[2] ———, *Consistent estimates of the parameters of a linear system*, Ann. Math. Statist., 40 (1969), pp. 2064–2075.

[3] ———, *Positive solutions to $X = A - BX^{-1}B^*$*, Linear Algebra Appl., 134 (1990), pp. 53–62.

[4] T. ANDO, *Limit of cascade iteration of matrices*, Numer. Funct. Anal. Optim., 21 (1980), pp. 579–589.

[5] A. BEN-ISRAEL, *A note on an iterative method for generalized inversion of matrices*, Math. Comp., 20 (1966), pp. 439–440.

[6] R. S. BUCY, *A priori bound for the Riccati equation*, in Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Vol. III: Probability Theory, Univ. of California Press, Berkeley, 1972, pp. 645–656.

[7] J. C. ENGWERDA, *On the existence of a positive definite solution of the matrix equation $X + A^T X^{-1} A = I$*, Linear Algebra Appl., 194 (1993), pp. 91–108.

[8] J. C. ENGWERDA, A. C. M. RAN, AND A. L. RIJKEBOER, *Necessary and sufficient conditions for the existence of a positive definite solution of the matrix equation $X + A^* X^{-1} A = Q$*, Linear Algebra Appl., 186 (1993), pp. 255–275.

[9] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, Johns Hopkins Univ. Press, Baltimore, MD, 1989.

[10] W. L. GREEN AND E. KAMEN, *Stabilization of linear systems over a commutative normed algebra with applications to spatially distributed parameter dependent systems*, SIAM J. Control Optim., 23 (1985), pp. 1–18.

[11] N. J. HIGHAM, *Computing a nearest symmetric positive semidefinite matrix*, Linear Algebra Appl., 103 (1988), pp. 103–118.

[12] N. J. HIGHAM AND R. S. SCHREIBER, *Fast polar decomposition of an arbitrary matrix*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 648–655.

[13] D. V. OUELLETTE, *Schur complements and statistics*, Linear Algebra Appl., 36 (1981), pp. 187–295.

[14] W. PUSZ AND S. L. WORONOWITZ, *Functional calculus for sequilinear forms and the purification map*, Rep. Math. Phys., 8 (1975), pp. 159–170.

[15] G. SCHULZ, *Iterative berechnung der reziproken matrix*, Z. Angew. Math. Mech., 13 (1933), pp. 57–59.

[16] G. E. TRAPP, *The Ricatti equation and the geometric mean*, Contemp. Math., 47 (1985), pp. 437–445.

[17] J. ZABEZYK, *Remarks on the control of discrete time distributed parameter systems*, SIAM J. Control Optim., 12 (1974), pp. 721–735.

[18] J. ZEMANIAN, *Non-uniform semi-infinite grounded grids*, SIAM J. Appl. Math., 13 (1982), pp. 770–788.

[19] X. ZHAN AND J. XIE, *On the matrix equation $X + A^T X^{-1} A = I$*, Linear Algebra Appl., to appear.

SIAM J. SCI. COMPUT.
Vol. 17, No. 5, pp. 1175–1201, September 1996

010

A SCHWARZ ALTERNATING PROCEDURE FOR SINGULAR PERTURBATION PROBLEMS*

MARC GARBEY†

Abstract. We show that the Schwarz alternating procedure offers a good algorithm for the numerical solution of singular perturbation problems, provided the domain decomposition is properly designed to resolve the boundary and transition layers. We give sharp estimates for the optimal position of the domain boundaries and study the convergence rates of the algorithm for various linear second-order singular perturbation problems. We report on implementation results for a turning-point problem and a combustion problem.

Key words. singular perturbations, boundary layers, domain decomposition

AMS subject classifications. Primary, 35B25, 65N05; Secondary, 35J25, 35J40

1. Introduction. The goal of this paper is to show that a properly designed Schwarz alternating procedure [5] offers an effective algorithm to solve singular perturbation problems. In particular, we show that a multiple-domain method can be *more accurate* and *more efficient* than a single-domain method—at least in the context of singular perturbation problems where the domain decomposition can be justified on the basis of physical arguments.

Since the First International Symposium on Domain Decomposition [16], much work has been done on domain decomposition methods, especially from the perspective of numerical linear algebra. However, few studies have been concerned with domain decomposition methods that are motivated by the physics of the phenomenon under investigation or by the properties of the underlying partial differential equations (PDEs); see [29, 7, 17, 31] and the references cited there.

In multiple-scale problems [24, 21], one can clearly take advantage of the special structure of a solution in the design of a domain decomposition method, as was first shown in the work of Chin [6]. In [19, 20], we began a systematic investigation of how to apply results of asymptotic analysis in the numerical computation of multiple-scale problems. As discussed in [11, 12], there are basically two approaches: one can start from a known asymptotic expansion and devise a numerical algorithm to compute the expansion, or one can start from a robust domain decomposition method and adjust the decomposition on the basis of asymptotic criteria. The former approach is very interesting but has several drawbacks: the algorithm cannot be a direct implementation of the method of matched asymptotic expansion [9, 4, 14] and, furthermore, the implementation in two or three space dimensions is not easy and can be compared to the difficulties encountered implementing a sophisticated method of front tracking.

In this paper we are concerned with the latter approach. We have chosen to start from the well-known Schwarz iterative procedure [18, 23, 26, 27, 30, 32] with *Dirichlet boundary conditions* for the sake of simplicity. We suppose that we have some knowledge of the position and thickness of boundary and transition layers [1] and derive accurate representations of the interface positions, which give the optimal numerical accuracy of the domain decomposition. We also propose a practical way to implement an adaptive domain decomposition method. With adaptivity it is not necessary to know accurately in advance the representation of the layers and the algorithm converges to the optimal interface(s) position(s).

For specific singular perturbation problems, we compute the convergence rate of the Schwarz alternating procedure *as a function of the small perturbation parameter*. We show

*Received by the editors October 15, 1993; accepted for publication (in revised form) February 25, 1995. This research was supported by Region Rhone Alpes-France.

†LAN, Université Claude Bernard Lyon 1, 69622 Villeurbanne cedex, France (garbey@lan1.univ-lyon1.fr).

that the Schwarz alternating procedure converges in a few iterations, particularly when the problem is convection-dominated. This last result is related, for example, to the work of Kuznetsov [22], who showed that the time step for a parabolic problem can be considered as a small perturbation parameter. However, Kuznetsov did not consider the singular perturbation problem.

An advantage of the approach under discussion is that the computation of more realistic problems is relatively easy to implement, as it is based on the improvement of an already existing technique. Also, one may want to solve irregular structures, such as layers, and keep the discretization grid regular in each subdomain (as opposed to taking an adaptive mesh without domain decomposition). This advantage is important for parallel computing; one can still use different numerical schemes in different subdomains.

For problems in two space dimensions, we show that a combination of the alternating direction iteration (ADI) and the Schwarz alternating procedure offers a good option for the accurate resolution of an inner layer; ADI is used as a massively parallel algorithm [10] to solve a problem that is smooth in the outer as well as the inner domain because of the stretching of the layer. Also, the coupling between the subdomain computations can be weak because ADI is implicit only in one space direction. We have implemented the Schwarz alternating procedure for a combustion problem to illustrate the potential of the method.

In this paper, we have not yet taken full advantage of the asymptotic results since the numerical scheme is basically the same on each subdomain. Most often, the asymptotic results suggest a heterogeneous domain decomposition [29]. We have recently focused our work in this direction [13]. Our choice of *Dirichlet boundary conditions* for the Schwarz alternating procedure is then useful to efficiently implement the so-called (numerical) matching between different approximations in the overlapping regions.

The plan of this paper is as follows. In §2 we compute the optimal interface position for the solution of an ordinary layer and compute the convergence rate of the Schwarz alternating procedure for different linear elliptic singular perturbation problems in one dimension. In §3 we implement an adaptive Schwarz domain decomposition to solve the inner layer of a turning-point problem. In §4 we test our technique on a two-dimensional turning-point problem; we solve the inner layer with a combination of Schwarz domain decomposition and ADI and illustrate the efficiency of the method. In §5 we solve a combustion problem describing a cellular flame; this problem is highly nonlinear and difficult to solve [2].

2. Boundary layers in one dimension. In this section we consider linear second-order singular perturbation problems of the following type:

$$L_\varepsilon \Phi \equiv \varepsilon L_1 \Phi + L_0 \Phi = F \text{ on } \Omega = (0, 1); \quad \Phi(0) = \alpha_0, \;\; \Phi(1) = \alpha_1. \tag{2.1}$$

Here, $L_1 \equiv \left(d^2/dx^2\right)$, $L_0 \equiv \beta(x, \varepsilon)(d/dx) + \gamma(x, \varepsilon)$, and $F \equiv F(x, \varepsilon)$; ε is a small positive parameter, $\varepsilon \in (0, \varepsilon_0]$ for some $\varepsilon_0 > 0$. The coefficients β, γ, and the function F are infinitely differentiable on $[0, 1]$.

Problems of this type exhibit boundary layers and transition layers; cf. [24]. We restrict ourselves to the case of a single boundary layer in the neighborhood of 0.

The Schwarz alternating procedure for problems of the type (2.1) is described in [5]. First, the domain Ω is split into two overlapping subdomains, $\Omega_{\text{inner}} = [0, a]$ and $\Omega_{\text{outer}} = [b, 1]$, with $a > b$; Ω_{inner} covers the boundary layer at 0, Ω_{outer} represents the domain of validity of the outer approximation. On each subdomain, the differential expressions are discretized by means of a finite difference scheme.

We assume that the mesh on each subdomain is regular; h_1 is the mesh size on Ω_{outer}, h_2 the mesh size on Ω_{inner}. If L^{h1} and L^{h2} are the corresponding discretized operators, then the iterative scheme is

$$L^{h1}\Phi^{p}_{\text{outer}} = F \text{ on } \Omega_{\text{outer}}, \quad \Phi^{p}_{\text{outer}}(b) = \Phi^{p}_{\text{inner}}(b), \ \Phi^{p}_{\text{outer}}(1) = \alpha_1, \tag{2.2}$$

$$L^{h2}\Phi^{p+1}_{\text{inner}} = F \text{ on } \Omega_{\text{inner}}, \quad \Phi^{p+1}_{\text{inner}}(0) = \alpha_0, \ \Phi^{p+1}_{\text{inner}}(a) = \Phi^{p}_{\text{outer}}(a), \tag{2.3}$$

for $p = 0, 1, \dots$. To start the scheme, one imposes an artificial boundary condition at point b.

The method is readily generalized to an arbitrary number of subdomains $\Omega_i = [b_i, a_i]$, where $0 = b_N < b_{N-1} < a_N < b_{N-2} < a_{N-1} < \cdots < b_0 < a_1 < a_0 = 1$. For example, the case of three subdomains leads to an iterative scheme with two possibly decoupled subproblems:

$$L^{h_0}\Phi^{p}_{0} = F \text{ on } \Omega_0, \quad \Phi^{p}_{0}(b_0) = \Phi^{p}_{1}(b_0), \ \Phi^{p}_{0}(1) = \alpha_1,$$

$$L^{h_2}\Phi^{p}_{2} = F \text{ on } \Omega_2, \quad \Phi^{p}_{2}(0) = \alpha_0, \ \Phi^{p}_{2}(a_2) = \Phi^{p}_{1}(a_2),$$

followed by

$$L^{h_1}\Phi^{p+1}_{1} = F \text{ on } \Omega_1, \quad \Phi^{p+1}_{1}(b_1) = \Phi^{p}_{0}(b_1), \ \Phi^{p+1}_{1}(a_1) = \Phi^{p}_{0}(a_1).$$

Artificial boundary conditions at the pair (a_2, b_0) are imposed to start the scheme. To solve a problem with a boundary layer at 0, one uses Ω_2 as an inner domain and Ω_0 as an outer domain; the intermediate domain Ω_1 links the inner and outer domains. A priori, we have $h_0(\varepsilon) \succ\succ h_1(\varepsilon) \succ\succ h_2(\varepsilon)$.

We restrict ourselves to use the same finite difference scheme in each subdomain with *Dirichlet boundary conditions*. For an arbitrary domain D with a mesh size parametrized by h, we suppose that the scheme must be such that the estimate

$$\max |\psi - \psi^h| \sim C\delta^q h^n \max \left| \frac{d^{n+p}\psi}{dx^{n+p}} \right| \tag{2.4}$$

is satisfied for some set of integers p, q, and n. Here, ψ is the exact solution of the Dirichlet problem in domain D and ψ^h its discrete approximation; the constant δ is a measure of the size of the domain D and C a generic positive constant or possibly an order function of ϵ, which does not depend on h. The symbol $\sim$ ($\approx$) denotes strong (weak) asymptotic equivalence; see [3]. By way of example, a centered second-order finite difference scheme applied to $-u'' = f$ gives $p = q = n = 2$.

2.1. Interface positions. First, we wish to determine the optimal interface positions, a_i and b_i, which minimize the error. We assume that the number of mesh points is the same, N, in each subdomain and consider only the case where the subgrids coincide at their endpoints, so no interpolation is needed in the iterative Schwarz procedure.

Let us start with a two-subdomains decomposition; we consider in this section the following decoupled subproblems:

$$L^{h1}\Phi^{h_1}_{\text{outer}} = F \text{ on } \Omega_{\text{outer}}, \quad \Phi^{h_1}_{\text{outer}}(b) = \Phi(b), \ \Phi^{h_1}_{\text{outer}}(1) = \alpha_1, \tag{2.5}$$

$$L^{h2}\Phi^{h_2}_{\text{inner}} = F \text{ on } \Omega_{\text{inner}}, \quad \Phi^{h_2}_{\text{inner}}(0) = \alpha_0; \ \Phi^{h_2}_{\text{inner}}(a) = \Phi(a), \tag{2.6}$$

where Φ is the exact solution of (2.1). Let Φ_{outer} (resp., Φ_{inner}) the restriction of Φ to Ω_{outer} (resp., Ω_{inner}). We define the following errors:

$$e_{\text{outer}} = \max_{\Omega_{\text{outer}}} |\Phi - \Phi^{h1}_{\text{outer}}| \tag{2.7}$$

and

$$e_{\text{inner}} = \max_{\Omega_{\text{inner}}} |\Phi - \Phi^{h2}_{\text{inner}}|. \tag{2.8}$$

According to the hypothesis on the finite difference discretization, we have

$$e_{\text{outer}} \sim C h_1^n \max \left| \frac{d^{n+p}\Phi_{\text{outer}}}{dx^{n+p}} \right| \tag{2.9}$$

and

$$e_{\text{inner}} \sim C a^q h_2^n \max \left| \frac{d^{n+p}\Phi_{\text{inner}}}{dx^{n+p}} \right| . \tag{2.10}$$

Our criteria to define the best interface positions (a, b) is that $\max(e_{\text{outer}}, e_{\text{inner}})$ should be asymptotically minimum.

Theorems 1 and 2 in §2.2 will show for some specific operators that this criterion is correct; i.e., we do not need to consider the artificial error introduced at the boundaries in the Schwarz alternating procedure to properly choose the interface positions.

The optimal position of the interface defined above now depends only on the property of the solution that we want to approximate. We restrict our analysis to functions that exhibit boundary layer phenomena as it can be the case for solutions of (2.1). But the following estimates on the interface position are independent of the operator. Let Φ_0 be the outer expansion of Φ. The main hypothesis is that we have some a priori knowledge of the *qualitative behavior* of the corrector $\Theta(x, \varepsilon) = \Phi(x, \varepsilon) - \Phi_0(x, \varepsilon)$ in the boundary layer. In the case of two subdomains, we have the following result.

LEMMA 1. *Suppose that*

(i) *the corrector function Θ is strictly of order one,*

(ii) *the inner variable is $\xi = x/\varepsilon$ and all derivatives of $\tilde{\Theta}(\xi, \varepsilon) = \Theta(x, \varepsilon)$ with respect to ξ are of order one,*

(iii) *the corrector function Θ is exponentially decreasing, $\tilde{\Theta}(\xi, \varepsilon) \sim C \exp(-C_0\xi)$ as $\xi \to \infty$.*

Then the error in each subdomain is minimum when the overlap is minimum. For any fixed overlap $(a - b = k_0 h_1)$, the accuracy is optimal in both subdomains if

$$a^{n+q} \sim C \exp\left(-(C_0/\varepsilon)(a - k_0/N)\right).$$

Proof. The error in the outer domain is

$$E^1 = \max_{\Omega_{\text{outer}}} |\Phi - \Phi^{h1}_{\text{outer}}| \sim C h_1^n \max_{\Omega_{\text{outer}}} \left| \frac{d^{n+p}\Phi}{dx^{n+p}} \right| \sim C \exp(-C_0 b/\varepsilon) h_1^n \varepsilon^{-(n+p)}.$$

Similarly, the error in the inner domain is

$$E^2 = \max_{\Omega_{\text{inner}}} |\Phi - \Phi^{h2}_{\text{inner}}| \sim C a^q h_2^n \max_{\Omega_{\text{inner}}} \left| \frac{d^{n+p}\Phi}{dx^{n+p}} \right| \sim C h_2^n a^q \varepsilon^{-(n+p)}.$$

If we impose a fixed overlap, $a - b = k_0 h_1$, where k_0 is an integer of order one (possibly equal to one), then

$$E^1 \sim C \exp\left(-(C_0/\varepsilon)(a - k_0/N)\right) N^{-n} \varepsilon^{-(n+p)}$$

and

$$E^2 \sim C a^{n+q} N^{-n} \varepsilon^{-(n+p)}.$$

Hence, if ε is sufficiently small, then E^2 is an increasing function and E^1 a decreasing function of a. The accuracy is optimal in both subdomains if $E^2 \sim E^1$, which is the case if $a^{n+q} \sim C \exp(-(C_0/\varepsilon)(a - k_0/N))$.

The asymptotic relation above shows that the optimal position of the interface a is an increasing function of k_0. Therefore, the optimal error E^2 is an increasing function of k_0 as well as E^1 and the error in both domains is minimum when the overlap is minimum that is $k_0 = 1$. □

LEMMA 2. *Suppose that the conditions of Lemma* 1 *are satisfied and that, in addition,* $b \sim Ca$. *Then, for any fixed overlap, the maximum of the error in each subdomain is minimum if*

$$b = (n + q + o(1))(\varepsilon/C_0)\log(C_0/\varepsilon)$$

and, consequently,

$$a \approx \varepsilon \log(\varepsilon^{-1}).$$

Proof. The maximum of the error in both subdomains is minimum when $e_{\text{inner}} \sim e_{\text{outer}}$, which is the case if $a^{n+q} \sim C \exp(-C_0 b/\varepsilon)$. If $b \sim Ca$, we obtain $b^k \sim C \exp(-b/\varepsilon_1)$, where $k = n + q$ and $\varepsilon_1 \sim \varepsilon/C_0$.

We look for b of the form $b = \eta(\varepsilon_1)\varepsilon_1 \log \varepsilon_1^{-1}$, where η is a positive function of order one. Then $b^k \sim C\varepsilon_1^k \log^k \varepsilon_1^{-1}$ and $\exp(-b/\varepsilon_1) \sim \varepsilon_1^\eta$. Therefore, the quantity $b^k/\exp(-b/\varepsilon_1)$ is very sensitive to the choice of the function $\eta(\varepsilon_1)$. We obtain

$$(k - \nu)\varepsilon_1 \log\left(\varepsilon_1^{-1}\right) < b < k\varepsilon_1 \log\left(\varepsilon_1^{-1}\right)$$

for all $\nu > 0$. More precisely, we look for b of the form $b = (k - \delta(\varepsilon_1))\varepsilon_1 \log\left(\varepsilon_1^{-1}\right)$, where $\delta \prec\prec 1$. The condition $b^{n+q} \sim \exp(-C_0 b/\varepsilon)$ is satisfied if and only if $\delta(\varepsilon_1) \sim (n+q)\log\left(\log\left(\varepsilon_1^{-1}\right)\right)/\log\left(\varepsilon_1^{-1}\right)$. As $a \approx b$, we conclude that $a \approx \varepsilon \log\left(\varepsilon_1^{-1}\right)$. □

Note that the hypothesis $b \sim Ca$ is satisfied when $N \geq 1/(C\varepsilon \log(\varepsilon/C_0)^{-1})$.

Remark 1. The estimate of Lemma 2 shows that the size of the inner subdomain Ω_{inner} increases with the order of the scheme and with the size of the overlap. It is interesting to note that the thickness of the layer obtained on the basis of the previous criterion is not simply ε but involves logarithmic terms, as in [1].

Remark 2. The estimate on b in Lemma 2 is numerically accurate only for very small values of ε. For moderate values of ε, one should use the next higher-order approximation of b, including the order function $\delta(\varepsilon)$.

Lemma 2 can be generalized to an arbitrary number of subdomains; the numerical error in each subdomain $\Omega_j = [b_j, a_j]$ is

$$E_j \sim C \exp(-C_0 b_j/\varepsilon) a_j^q h_j^n \varepsilon^{-(n+p)}.$$

We will show that the quantity $\max_{j=0,\ldots,N}(E_j)$ is minimum when the sequence of interfaces (b_j) satisfies the set of asymptotic relations $E_j \sim E_{j+1}$ for $j = 0, \ldots, N - 1$, that is, when the numerical error in each subdomain is of the same order.

The computation of the sequence of interfaces that satisfies the nonlinear problem $E_j = E_{j+1}$ for $j = 0, \ldots, N - 1$ can be done numerically for a given ε. However, we can obtain an asymptotic approximation of the b_j.

LEMMA 3. *Suppose we have* $N + 1$ *overlapping subdomains,* $\Omega_j = [b_j, a_j]$, *where* $0 = b_N < b_{N-1} < a_N < b_{N-2} < a_{N-1} < \cdots < b_0 < a_1 < a_0 = 1$, *and that* $a_j \sim b_{j-1}$ *for* $j = 1, \ldots, N$. *The maximum of the error in each subdomain will be minimum when*

$$b_j = (n + q + o(1))\varepsilon \log_{j+1}(C_0/\varepsilon).$$

Here, $\log_j$ *stands for* $\log(\log(\log\ldots))$, *j times.*

Proof. Let $k = n + q$ and $\varepsilon_1 \sim \varepsilon/C_0$. Since $a_j \sim b_{j-1}$, the error in each subdomain is $E_j \sim C\exp\left(-b_j/\varepsilon_1\right) b_{j-1}^k \varepsilon^{-(n+p)}$. For any j, E_j is a decreasing function of b_j and E_{j+1} is an increasing function of b_j. Therefore, by induction, $\max_{j=1,\dots,N}(E_j)$ is minimum when $E_j = E_{j+1}$ for $j = 0, \dots, N-1$.

For any function b_{N-1}, one explicitly gets a set of functions $(b_j)_{j=0,\dots,N-2}$ which satisfy the relation $E_j \sim E_{j+1}$ for $j = 1, \dots, N-1$. The corresponding b_j, and in particular b_0, are increasing functions of b_{N-1}. Therefore, E_0 is a decreasing function of b_{N-1} under the constraint $E_j = E_{j+1}$ for $j = 0, \dots, N-1$ and E_N is an increasing function of b_{N-1}. The existence and uniqueness of b_{N-1} such that $E_0 = E_N$ then follow from the intermediate variable theorem.

Next, we will derive an asymptotic estimate for the interface position in the case of three subdomains. The error in each subdomain balances when

$$\exp(-b_0/\varepsilon_1) \sim C b_0^k \exp(-b_1/\varepsilon_1) \sim C b_1^k.$$

We look for b_0 in the form $b_0 = (k - \delta(\varepsilon))\varepsilon_1 \log(\varepsilon_1^{-1})$, where $\delta \prec\prec 1$. As $\exp(-b_0/\varepsilon_1) \sim \varepsilon_1^{k-\delta}$, we have $b_1 \sim C\varepsilon_1^{1-\delta/k}$. Therefore, δ must satisfy the asymptotic relation $\varepsilon_1^{k-\delta} \sim C\varepsilon_1^k \log^k(\varepsilon_1^{-1}) \exp(-\varepsilon_1^{-\delta/k})$. Using the fact that

$$\frac{\log(\log^k(\log^k \varepsilon_1^{-1}))}{\log \varepsilon_1^{-1}} > \delta > \frac{\log(\log^k(\log^{k-\nu} \varepsilon_1^{-1}))}{\log \varepsilon_1^{-1}},$$

for all $\nu > 0$, we conclude that $b_1 \sim k\varepsilon_1 \log(\log(\varepsilon_1^{-1}))$. Thus, in the case of three subdomains,

$$b_0 = (k + o(1))\varepsilon_1 \log(\varepsilon_1^{-1}), \quad b_1 = (k + o(1))\varepsilon_1 \log(\log(\varepsilon_1^{-1})).$$

The proof for an arbitrary number of subdomains proceeds along the same lines. □

Remark 3. All previous results are still valid for nonoverlapping subdomains. When one solves a boundary layer at 0, the numerical accuracy in each subdomain is dominated by the error in the neighborhood of its left boundary.

2.2. Damping of artificial boundary errors and convergence. Let $[A, B] \subset [0, 1]$ be a given subdomain of Ω used in a Schwarz iterative procedure. The convergence and accuracy of the Schwarz method essentially depend on the way an error introduced at one of the artificial boundary points propagates inside the subdomain.

Following Cai and McCormick [5], we consider the linear problem

$$-u'' = F, \quad u(0) = \alpha_0, \quad u(1) = \alpha_1, \tag{2.11}$$

with the usual three-point discretization applied to each subdomain $[A, B]$. Let $e_i = u_i - u(x_i)$ be the error on the regular grid $\{x_i = A + i(B - A)/N : i = 0, \dots, N\}$. Suppose that $e_0 = 0$, and let e_N be the error at the artificial boundary B. We have $e_i = (i/N)e_N$ for $i = 0, \dots, N$. In other words, any error at the artificial boundary B decreases linearly inside the subdomain $[A, B]$. The situation is analogous for the damping of an error introduced at the artificial boundary A. This linear convergence is the key reason for the poor efficiency of the *straightforward Schwarz alternating procedure with Dirichlet boundary conditions* for the elliptic operator $L[u] = -\Delta u$ demonstrated in [5]. Of course, the situation is different for the positive operator $L[u] = -\Delta u + \gamma u$ with $\gamma > 0$. Also, for regular problems (as opposed to singular perturbation problems), it is well known that more sophisticated boundary conditions can lead to a very efficient Schwarz method; see [32], for example. We will show that the situation is considerably different for a singular perturbation problem of elliptic-type and that we may, in fact, have fast convergence with relatively small overlap even if we restrict ourselves to *Dirichlet boundary conditions.*

Let $[b_0, a_0]$ and $[b_1, a_1]$ be the two overlapping subdomains of a domain decomposition, with $b_1 < b_0 < a_1 < a_0$. We consider the homogeneous discretized problems

$$L^{h_1}[e_1] = 0, e_1(b_0) = \alpha_1,\ e_1(a_0) = 0, \tag{2.12}$$

$$L^{h_2}[e_2] = 0, e_2(b_1) = 0,\ e_2(a_1) = \alpha_2. \tag{2.13}$$

We will say that the real numbers $\zeta_{\rightarrow}$ and $\zeta_{\leftarrow}$ are *damping factors* if $e_2(a_1) \leq \zeta_{\rightarrow} e_2(b_0)$ and $e_1(b_0) \leq \zeta_{\leftarrow} e_1(a_1)$.

An iteration of the Schwarz method on a domain decomposition that includes the previous two particular overlapping subdomains results in $e_1^{p+1}(b_0) \leq \zeta_{\rightarrow}\zeta_{\leftarrow} e_1^p(b_0)$ and $e_2^{p+1}(a_1) \leq \zeta_{\rightarrow}\zeta_{\leftarrow} e_2^p(a_1)$. The convergence of the scheme follows when the discretized operator satisfies a maximum principle and the product of the damping factors is less than one.

The damping factors can be computed for various singular perturbation problems and finite difference schemes.

2.2.1. The case $L[u] = -\varepsilon u'' + \gamma(x)\, u = F$, $\gamma > 0$. First, we solve a boundary layer at 0 with a two-subdomain Schwarz procedure and the usual three-point discretization. For the singular perturbation problem $-\varepsilon u'' + \gamma u = F$ on $[0, 1]$, the thickness of the boundary layer at 0 is $\varepsilon^{1/2}$. The inner variable is $\xi = x/\varepsilon^{1/2}$, and the layer can be resolved with a corrector function $\tilde{\Theta}$ which satisfies the asymptotic relation $\tilde{\Theta}(\xi, \varepsilon) \sim C \exp(-C_0\xi)$ as $\xi \rightarrow \infty$. Furthermore, $C_0 = \gamma^{1/2}(0)$. It follows from Lemma 2 that $\max(e_{\text{outer}}, e_{\text{inner}})$ defined by (2.7) and (2.8) is minimum if

$$a \approx \varepsilon^{1/2} \log \varepsilon^{-1}, \quad N \geq C/\left(\varepsilon^{1/2} \log \varepsilon^{-1}\right).$$

LEMMA 4. *Let $[0, a]$ and $[b, 1]$ be two overlapping subdomains, with $a - b = k_0 h_1$ and $a \sim b \approx \varepsilon^{1/2} \log \varepsilon^{-1}$.*

If $N \sim 1/\left(\varepsilon^{1/2} \log \varepsilon^{-1}\right)$, then

$$\zeta_{\rightarrow} \sim \log^{-2k_0} \varepsilon^{-1}, \quad \zeta_{\leftarrow} \sim \varepsilon^{k_0}.$$

If $N \sim C^{-1}(\gamma(0))/\varepsilon)^{1/2}$, then

$$\zeta_{\rightarrow} \sim R^{-k_0}, \quad \zeta_{\leftarrow} \sim \exp(-k_0 C),$$

where $R = 1 + \frac{1}{2}C^2 + C(1 + \frac{1}{4}C^2)^{1/2}$.

Proof. We compute the damping factor by applying the three-point discretization in each subdomain to the homogeneous problem

$$-\varepsilon(e_{i+1} - 2e_i + e_{i-1}) + h^2 \gamma_i e_i = 0.$$

The finite difference equation can be written as

$$e_{i+1} - (2 + \gamma_i h^2/\varepsilon) e_i + e_{i-1} = 0.$$

Then

$$\zeta_{\leftarrow} \sim (R(h_2, \varepsilon))^{-k_0 h_1/h_2}, \quad \zeta_{\rightarrow} \sim (R(h_1, \varepsilon))^{-k_0},$$

where $R(h, \varepsilon)$ is the larger root of the quadratic polynomial $R^2 - (2 + \gamma(0)h^2/\varepsilon)R + 1$,

$$R(h, \varepsilon) = 1 + 1/2(h^2\gamma(0)/\varepsilon) + (h\gamma(0)^{1/2}/\varepsilon^{1/2})\left(1 + h^2\gamma(0)/(4\varepsilon)\right)^{1/2}.$$

The lemma follows upon substitution of the expression of R. □

We now prove the convergence of the method using the result of Lemma 3 and show that the criterion to locate the best interface position defined in §2.1 is correct.

THEOREM 1. *Let* $\zeta = \zeta_{\leftarrow} \times \zeta_{\rightarrow}$, *and let* Φ *be the solution of the Dirichlet problem*

$$L[u] = -\varepsilon u'' + \gamma(x)u = F, \quad u(0) = \alpha_0, \; u(1) = \alpha_1. \tag{2.14}$$

Let Φ^p_{outer} *and* Φ^p_{inner} *be the solutions of the iterative scheme*

$$L^{h1}\Phi^p_{\text{outer}} = F \text{ on } \Omega_{\text{outer}}, \quad \Phi^p_{\text{outer}}(b) = \Phi^p_{\text{inner}}(b), \quad \Phi^p_{\text{outer}}(1) = \alpha_1, \tag{2.15}$$

$$L^{h2}\Phi^{p+1}_{\text{inner}} = F \text{ on } \Omega_{\text{inner}}, \quad \Phi^{p+1}_{\text{inner}}(0) = \alpha_0, \quad \Phi^{p+1}_{\text{inner}}(a) = \Phi^p_{\text{outer}}(a), \tag{2.16}$$

where

$$L^{h_i}[u] = -\varepsilon\frac{u_{i+1} - 2u_i + u_{i-1}}{h_i^2} + \gamma(x_i)u^i.$$

Let

$$\Phi^p = \begin{cases} \Phi^p_{\text{inner}} & \text{on } M_{\text{inner}} = \{x_i = i(a/N), i = 0, \dots, N\}, \\ \Phi^p_{\text{outer}} & \text{on } M_{\text{outer}} = \{y_i = b + i\frac{1-b}{N}, i = 0, \dots, N, \; y_i \in \Omega_{\text{outer}}\backslash\Omega_{\text{inner}}\}. \end{cases}$$

Let e_{outer} *and* e_{inner} *be the numerical errors in the solution of the Dirichlet subproblems defined by* (2.5), (2.7), (2.6), *and* (2.8). *Let* $\|\cdot\|_\infty$ *be the maximum norm on the composite grid* $M_{\text{outer}} \bigcup M_{\text{inner}}$. *Then*

$$\|\Phi - \Phi_p\|_\infty \leq C(\zeta^p + \max(e_{\text{outer}}, e_{\text{inner}})).$$

Proof. Let $\tilde{\Phi} = (\tilde{\Phi}_{i,j})_{i=0..N, j=1,2}$ be the solution to the following linear system:

$$-\varepsilon\frac{\tilde{\Phi}_{i+1,1} - 2\tilde{\Phi}_{i,1} + \tilde{\Phi}_{i-1,1}}{h_1^2} + \gamma(y_i)\tilde{\Phi}_{i,1} = F(y_i), \quad i = 1, \dots, N-1, \tag{2.17}$$

$$-\varepsilon\frac{\tilde{\Phi}_{i+1,2} - 2\tilde{\Phi}_{i,2} + \tilde{\Phi}_{i-1,2}}{h_2^2} + \gamma(x_i)\tilde{\Phi}_{i,2} = F(x_i), \quad i = 1, \dots, N-1, \tag{2.18}$$

$$\tilde{\Phi}_{N,1} = \alpha_1, \; \tilde{\Phi}_{0,2} = \alpha_0, \tag{2.19}$$

$$\tilde{\Phi}_{0,1} = \tilde{\Phi}_{k,2}, \; \tilde{\Phi}_{N,2} = \tilde{\Phi}_{1,1}, \tag{2.20}$$

with $h_1 = \frac{1-b}{N}$, $h_2 = \frac{a}{N}$, $x_k = b$. Let Φ be the exact solution of the Dirichlet problem: $-\varepsilon u'' + \gamma(x)u = F$, $u(0) = \alpha_0, u(1) = \alpha_1$. Using a discrete maximum principle on the composite grid $M_{\text{inner}} \bigcap M_{\text{outer}}$, we have the existence and uniqueness of $\tilde{\Phi}$.

Let $e_{i,j} = \Phi_{i,j} - \tilde{\Phi}_{i,j}, i = 0, \dots, N, j = 1, 2$, where $\Phi_{i,j}$ is the trace of Φ on the composite grid $M_{\text{outer}} \bigcup M_{\text{inner}}$. We have

$$-\varepsilon\frac{e_{i+1,1} - 2e_{i,1} + e_{i-1,1}}{h_1^2} + \gamma(y_i)e_{i,1} = \epsilon\frac{h_1^2}{12}\Phi^{(4)}_{i,1}, \quad i = 1, \dots, N-1, \tag{2.21}$$

$$-\varepsilon\frac{e_{i+1,2} - 2e_{i,2} + e_{i-1,2}}{h_2^2} + \gamma(x_i)e_{i,2} = \epsilon\frac{h_2^2}{12}\Phi^{(4)}_{i,2}, \quad i = 1, \dots, N-1, \tag{2.22}$$

$$e_{N,1} = 0, \; e_{0,2} = O, \tag{2.23}$$

$$e_{0,1} = e_{k,2}, \; e_{N,2} = e_{1,1}, \tag{2.24}$$

where $\Phi^{(4)}_{i,j}$ denotes the fourth-order derivative of Φ obtained in the Taylor expansion. If $|e_{i,1}|$ is not maximal at the boundary $i = 0$, we have

$$\max_{i=0,\dots,N} |e_{i,1}| \leq C\epsilon h_1^2 \max(\Phi^{(4)}) \approx e_{\text{outer}},$$

otherwise, we have

$$|e_{N,2}| = |e_{1,1}| < |e_{k,2}| = |e_{0,1}|;$$

that is, $|e_{i,2}|$ cannot be maximal at the boundary $i = N$ and

$$\max_{i=0,\dots,N} |e_{i,2}| \leq C\epsilon a^2 h_2^2 \max_{\Phi^{(4)}} \approx e_{\text{inner}}.$$

Consequently, we have

$$\|\Phi - \tilde{\Phi}\|_\infty \leq C \max(e_{\text{outer}}, e_{\text{inner}}).$$

Let $\tilde{\Phi}_{\text{outer}} = (\tilde{\Phi}_{i,1})_{i=0,\dots,N}$ (resp., $\tilde{\Phi}_{\text{inner}} = (\tilde{\Phi}_{i,2})_{i=0,\dots,N}$) and $v^p_{\text{outer}} = \Phi^p_{\text{outer}} - \tilde{\Phi}_{\text{outer}}$ (resp., $v^p_{\text{inner}} = \Phi^p_{\text{inner}} - \tilde{\Phi}_{\text{inner}}$).

We have

$$L^{h1} v^p_{\text{outer}} = 0 \text{ on } \Omega_{\text{outer}}, \; v^p_{\text{outer}}(b) = v^p_{\text{inner}}(b), \; v^p_{\text{outer}}(1) = 0,$$

$$L^{h2} v^{p+1}_{\text{inner}} = 0 \text{ on } \Omega_{\text{inner}}, \; v^{p+1}_{\text{inner}}(a) = v^p_{\text{outer}}(a), \; v^{p+1}_{inner}(0) = 0.$$

We conclude from Lemma 4 that

$$\|v^p_{\text{outer}}\|_\infty \leq C\zeta^p \text{ on } M_{\text{outer}}$$

and

$$\|v^p_{\text{inner}}\|_\infty \leq C\zeta^p \text{ on } M_{\text{inner}}$$

and, therefore,

$$\|\Phi - \Phi^p\| \leq C\zeta^p + \max(e_{\text{outer}}, e_{\text{inner}}).$$

According to this estimation p must be chosen such that

$$\zeta^p \approx \max(e_{\text{outer}}, e_{\text{inner}})$$

and the interfaces a and b such that $\max(e_{\text{outer}}, e_{\text{inner}})$ is asymptotically minimum. □

It is interesting to see how the damping factors behave when the number of subdomains increases. First, consider the three-subdomain case with $a_1 \sim b_0$ and $a_2 \sim b_1$. We consider the optimal situation described in Lemma 3, for which $a_1 \approx \varepsilon^{1/2}\log(\varepsilon^{-1})$ and $a_2 \approx \varepsilon^{1/2}\log(\log(\varepsilon^{-1}))$. To minimize the damping factor, we choose the smallest order for N that is still compatible with our hypothesis on the overlapping

$$N \sim 1/\left(\varepsilon^{1/2}\log(\varepsilon^{-1})\right).$$

The damping factors $\zeta_{\rightarrow}^{0,1}$ and $\zeta_{\leftarrow}^{0,1}$ corresponding to the two overlapping subdomains $\Omega_0 = [b_0, a_0 = 1]$ and $\Omega_1 = [b_1, a_1]$ are as good as in Lemma 4: $\zeta_{\rightarrow}^{0,1} \approx \log^{-2k_0} \varepsilon^{-1}$, and $\zeta_{\leftarrow}^{0,1} \approx \varepsilon^{k_0}$. On the other hand, for the subsequent damping factors corresponding to the overlapping domains Ω_1 and Ω_2, we obtain $\zeta_{\rightarrow}^{1,2} \approx \zeta_{\leftarrow}^{1,2} \approx 1 - k_1 \varepsilon^{1/2} \log^2(\varepsilon^{-1})$. Hence, the three-subdomain Schwarz method may converge very slowly under the previous hypothesis. We observe that $\varepsilon^{1/2} \log^2(\varepsilon^{-1})$ is less than $1/2$ if and only if $\varepsilon < 0.000017$. In other words, the damping factors $\zeta_{\rightarrow}^{1,2}$ and $\zeta_{\leftarrow}^{1,2}$ can be practically small for ε larger than 10^{-4}.

Under the hypothesis of Lemma 3, one can show that the situation deteriorates when the number of subdomains increases. For $i > 0$, $1 - \zeta_{\rightarrow}^{i,i+1} \gg 1 - \zeta_{\rightarrow}^{i+1,i+2}$ and $1 - \zeta_{\leftarrow}^{i,i+1} \gg 1 - \zeta_{\leftarrow}^{i+1,i+2}$. In practice, N may be much smaller than $\varepsilon^{-1/2} \log^{-1}(\varepsilon^{-1})$. However, we have observed in our numerical experiments that the convergence deteriorates when the number of subdomains increases and the relative size of the overlap $(a_i - b_{i-1})/h_i$ stays fixed.

2.2.2. The case $L[u] = \varepsilon u'' + \beta(x)u' + \gamma(x)u = F$, $\beta > 0$. We suppose for simplicity that the real function β is strictly positive on Ω, so we do not consider turning-point problems in this section. Let us first consider a first-order finite difference scheme with the appropriate one-sided difference approximation of the convective term

$$\varepsilon \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + \beta(x_i) \frac{u_{i+1} - u_i}{h} + \gamma(x_i) u_i = F_i.$$

This scheme is monotone when γ is a negative function. We know that for the Dirichlet singular perturbation problem $\varepsilon u'' + \beta u' + \gamma u = F$ on $[0, 1]$, the thickness of the boundary layer at 0 is ε. The inner variable is $\xi = x/\varepsilon$, and the layer can be resolved with a corrector function $\tilde{\Theta}$ that satisfies the estimate $\tilde{\Theta}(\xi, \varepsilon) \sim C \exp(-C_0 \xi)$ as $\xi \to \infty$. Furthermore, $C_0 = \beta^{1/2}(0)$. Let N be such that

$$N \geq (C/\varepsilon) \log \varepsilon^{-1}.$$

According to Lemma 2, $\max(e_{\text{outer}}, e_{\text{inner}})$ is minimum when

$$a \approx \varepsilon \log \varepsilon^{-1}.$$

LEMMA 5. *Let* $[0, a]$ *and* $[b, 1]$ *be two overlapping subdomains, with* $a - b = k_0 h_1$ *and* $a \sim b \approx \varepsilon \log \varepsilon^{-1}$.

If $N \sim 1/\left(\varepsilon \log \varepsilon^{-1}\right)$, *then*

$$\zeta_{\rightarrow} \sim \beta(0)^{-k_0} \log^{-k_0} \varepsilon^{-1}, \quad \zeta_{\leftarrow} \sim 1 + k_0 \frac{\gamma(0)}{\beta(0)} \varepsilon \log \varepsilon^{-1}.$$

If $N \sim C\varepsilon^{-1}$, *then*

$$\zeta_{\rightarrow} \sim \frac{1}{(1 + \beta(0)C)^{k_0}}, \quad \zeta_{\leftarrow} \sim 1 + Ck_0 \frac{\gamma(0)}{\beta(0)} \varepsilon.$$

Proof. The proof is analogous to the proof of the previous lemma. The three-point discretization in any subdomain applied to the homogeneous problem satisfies

$$\varepsilon(e_{i+1} - 2e_i + e_{i-1}) + \beta_i h(e_{i+1} - e_i) + h^2 \gamma_i e_i = 0.$$

The asymptotic behavior of the damping factors follows from a straightforward computation. □

When the previous first-order scheme satisfies a maximum principle, we have Theorem 2.

THEOREM 2. *With the same notational convention as in Theorem* 1, *we have*

$$\|\Phi - \Phi_p\|_\infty \leq C\zeta^p + \max(e_{\text{outer}}, e_{\text{inner}}).$$

Remark 4. Even if $\gamma(0)$ is positive, the product $\zeta_\rightarrow\zeta_\leftarrow$ is less than one for ε sufficiently small. However, for singular perturbation problems with convection, where $\beta(0) \neq 0$, the rate of damping is less advantageous than in the case where $\beta = 0$.

The result of Lemma 5 can be extended to the three-subdomain case as before. The conclusion is the same; namely, the efficiency of the method deteriorates as the number of subdomains increases. With the notation of the previous section we have in the three-subdomain case

$$\zeta_\rightarrow^{1,2} \sim \zeta_\leftarrow^{1,2} \sim 1 - k_1\beta(0)\varepsilon\log^2(\varepsilon^{-1}).$$

Now, let us consider the central finite difference scheme

$$\varepsilon\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + \beta_i\frac{u_{i+1} - u_{i-1}}{2h} + \gamma_i u_i = 0.$$

In general, the scheme is not monotone. However, one can still do the asymptotic analysis of the damping factor.

LEMMA 6. *Let* $[0, a]$ *and* $[b, 1]$ *be two overlapping subdomains with* $a - b = k_0 h_1$ *and* $a \sim b \approx \varepsilon\log\varepsilon^{-1}$. *If* $N \sim 1/\left(\varepsilon\log\varepsilon^{-1}\right)$, *then*

$$\zeta_\rightarrow \sim 1 - \frac{4k_0}{\beta(0)\log\varepsilon^{-1}}, \quad \zeta_\leftarrow \sim 1 + k_0\frac{\gamma(0)}{\beta(0)}\varepsilon\log\varepsilon^{-1}.$$

If $N \sim C\varepsilon^{-1}$, *where* $C \neq \frac{1}{2}\beta(0)$, *then*

$$\zeta_\rightarrow \sim \left(\frac{1 - 2C/\beta(0)}{1 + 2C/\beta(0)}\right)^{k_0}, \quad \zeta_\leftarrow \sim 1 + Ck_0\frac{\gamma(0)}{\beta(0)}\varepsilon.$$

We notice that the Schwarz alternating procedure converges faster with the appropriate one-sided finite difference scheme than with the centered finite difference scheme.

2.2.3. Time-dependent case, $u_t = \varepsilon u_{xx} + \beta(x)u_x + \gamma(x)u$. Next, we consider an initial boundary value problem on $\Omega \times [0, T]$ with Dirichlet conditions

$$\frac{\partial u}{\partial t} = \varepsilon\frac{\partial^2 u}{\partial x^2} + \beta(x, t)\frac{\partial u}{\partial x} + \gamma(x, t)u.$$

If we use an implicit Euler scheme for the time marching

$$\frac{u^{n+1} - u^n}{\Delta t} = \varepsilon u_{xx}^{n+1} + \beta^{n+1}u_x^{n+1} + \gamma^{n+1}u^{n+1};$$

then

$$-\varepsilon\Delta T u_{xx}^{n+1} - \beta^{n+1}\Delta T u_x^{n+1} + (1 - \gamma^{n+1}\Delta T)u^{n+1} = u^n.$$

In other words, the time-marching scheme leads to the solution of a singular perturbation problem at each time step. The small parameter of this singular perturbation problem is even smaller than ε, as the time step is a priori a small number, but the structure of the boundary layers is independent of the time step. Also, the implicit scheme is appropriate if the time-dependent singular perturbation problem exhibits an initial layer.

Again, we distinguish the case where we have convection ($\beta \neq 0$) from the case where convection is absent ($\beta = 0$).

For steady problems without convection, we have seen in Lemma 4 that the efficiency of the alternating Schwarz procedure improves as the perturbation parameter decreases. As ε is multiplied by ΔT, where $\Delta T \ll 1$, we can expect the method to be especially efficient for the solution of time-dependent singular perturbation problems without convection.

The case $\beta = 0$ is trivial, so from now on we consider only nonzero convection.

Consider a Schwarz alternating procedure with two overlapping subdomains to solve the equation

$$-\varepsilon \Delta T u_{xx} - \beta \Delta T u_x + (1 - \gamma \Delta T) u = F.$$

For the one-sided difference scheme we obtain the following result.

LEMMA 7. *Let* $[0, a]$ *and* $[b, 1]$ *be two overlapping subdomains, with* $a - b = k_0 h_1$ *and* $a \sim b \approx \varepsilon \log \varepsilon^{-1}$. *Let* δ *be an order function such that* $\delta \succ\succ 1$. *Let* N *be such that* $N \sim \varepsilon^{-1}\delta^{-1}$.

If $\Delta T \succ\succ \varepsilon\delta^2$, *then*

$$\zeta_{\rightarrow}\zeta_{\leftarrow} \sim (\beta\delta)^{-k_0}.$$

Proof. The proof of this lemma is a straightforward computation. □

From the asymptotic approximation of the damping factors given above we observe that the efficiency of the method improves compared with the steady case. Also, very few iterations of the Schwarz alternating procedure are required since the initial artificial boundary condition can be the trace of the solution at the previous time step. The constraint on the time step is reasonable, as the accuracy of the spatial Schwarz alternating procedure must be of at least the same order as the accuracy of the temporal Euler scheme. However, one can use higher-order time differencing schemes and, therefore, a larger time step. For larger time steps, we are progressively back to the situation of the steady problems described by Lemmas 5 and 6.

2.3. Numerical experiments. We consider, first, a linear second-order singular perturbation problem

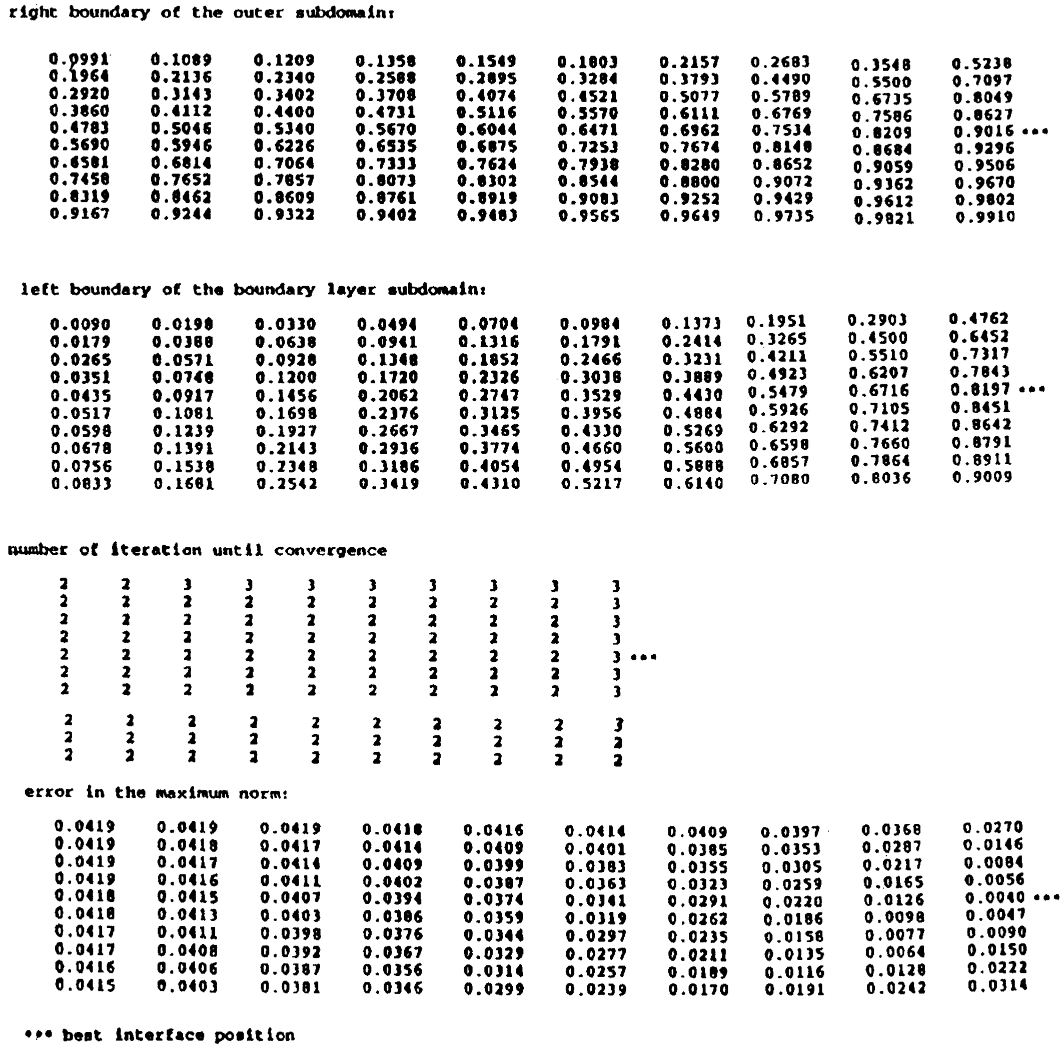
$$\varepsilon u'' + \beta u' + \gamma u = f$$

on the interval $\Omega = [0, 1]$, with a boundary layer located at the right end of the interval. To obtain a good numerical experiment, we choose f such that the outer expansion is nontrivial, so the reduced problem $\beta u' + \gamma u = f$ has a solution that is not constant. Furthermore, to simplify the numerics, we impose the condition that the grids of the subdomains coincide at the boundary points. For each experiment we use a known explicit solution, and any numerical error will be given in the discrete maximum norm on the composite grid, which is the union of the grids of the subdomains.

To balance the amount of work for each subproblem, we use the same number of points N for each subgrid. Then there is a finite number of possible sets of n subgrids that coincide at their end points. For example, in the two-subdomains case, the regular grids $\{x_i = ih_1 : i = 0, \dots, N\}$ of $\Omega_1 = [0, a]$ and $\{y_i = ih_2 + b : i = 0, \dots, N\}$ of $\Omega_2 = [b, 1]$, where $h_1 = a/N$ and $h_2 = (1 - b)/N$, $0 < b < a < 1$, coincide in a and b if and only if there exist $k_1, k_2 \in \{1, \dots, N - 1\}$, such that $a = k_1 h_2 + b$ and $b = k_2 h_1$. Hence, we have exactly $(N - 2)^2$ possible choices. A similar analysis for three subdomains gives $(N - 1)^4$ possible sets of interface positions.

TABLE 1

Error in the maximum norm and number of iterations with two subdomains for the operator $-\varepsilon u'' + \gamma u$. $-\varepsilon u'' + u = \cos(x)$, *two subdomains with* $N = 12$ *and* $\epsilon = 0.001$.



```
right boundary of the outer subdomain:

  0.0991   0.1089   0.1209   0.1358   0.1549   0.1803   0.2157   0.2683   0.3548   0.5238
  0.1964   0.2136   0.2340   0.2588   0.2895   0.3284   0.3793   0.4490   0.5500   0.7097
  0.2920   0.3143   0.3402   0.3708   0.4074   0.4521   0.5077   0.5789   0.6735   0.8049
  0.3860   0.4112   0.4400   0.4731   0.5116   0.5570   0.6111   0.6769   0.7586   0.8627
  0.4783   0.5046   0.5340   0.5670   0.6044   0.6471   0.6962   0.7534   0.8209   0.9016 •••
  0.5690   0.5946   0.6226   0.6535   0.6875   0.7253   0.7674   0.8148   0.8684   0.9296
  0.6581   0.6814   0.7064   0.7333   0.7624   0.7938   0.8280   0.8652   0.9059   0.9506
  0.7458   0.7652   0.7857   0.8073   0.8302   0.8544   0.8800   0.9072   0.9362   0.9670
  0.8319   0.8462   0.8609   0.8761   0.8919   0.9083   0.9252   0.9429   0.9612   0.9802
  0.9167   0.9244   0.9322   0.9402   0.9483   0.9565   0.9649   0.9735   0.9821   0.9910

 left boundary of the boundary layer subdomain:

  0.0090   0.0198   0.0330   0.0494   0.0704   0.0984   0.1373   0.1951   0.2903   0.4762
  0.0179   0.0388   0.0638   0.0941   0.1316   0.1791   0.2414   0.3265   0.4500   0.6452
  0.0265   0.0571   0.0928   0.1348   0.1852   0.2466   0.3231   0.4211   0.5510   0.7317
  0.0351   0.0748   0.1200   0.1720   0.2326   0.3038   0.3889   0.4923   0.6207   0.7843
  0.0435   0.0917   0.1456   0.2062   0.2747   0.3529   0.4430   0.5479   0.6716   0.8197 •••
  0.0517   0.1081   0.1698   0.2376   0.3125   0.3956   0.4884   0.5926   0.7105   0.8451
  0.0598   0.1239   0.1927   0.2667   0.3465   0.4330   0.5269   0.6292   0.7412   0.8642
  0.0678   0.1391   0.2143   0.2936   0.3774   0.4660   0.5600   0.6598   0.7660   0.8791
  0.0756   0.1538   0.2348   0.3186   0.4054   0.4954   0.5888   0.6857   0.7864   0.8911
  0.0833   0.1681   0.2542   0.3419   0.4310   0.5217   0.6140   0.7080   0.8036   0.9009

number of iteration until convergence

   2    2    3    3    3    3    3    3    3    3
   2    2    2    2    2    2    2    2    2    3
   2    2    2    2    2    2    2    2    2    3
   2    2    2    2    2    2    2    2    2    3
   2    2    2    2    2    2    2    2    2    3 •••
   2    2    2    2    2    2    2    2    2    3
   2    2    2    2    2    2    2    2    2    3

   2    2    2    2    2    2    2    2    2    3
   2    2    2    2    2    2    2    2    2    2
   2    2    2    2    2    2    2    2    2    2

 error in the maximum norm:

  0.0419   0.0419   0.0419   0.0418   0.0416   0.0414   0.0409   0.0397   0.0368   0.0270
  0.0419   0.0418   0.0417   0.0414   0.0409   0.0401   0.0385   0.0353   0.0287   0.0146
  0.0419   0.0417   0.0414   0.0409   0.0399   0.0383   0.0355   0.0305   0.0217   0.0084
  0.0419   0.0416   0.0411   0.0402   0.0387   0.0363   0.0323   0.0259   0.0165   0.0056
  0.0418   0.0415   0.0407   0.0394   0.0374   0.0341   0.0291   0.0220   0.0126   0.0040 •••
  0.0418   0.0413   0.0403   0.0386   0.0359   0.0319   0.0262   0.0186   0.0098   0.0047
  0.0417   0.0411   0.0398   0.0376   0.0344   0.0297   0.0235   0.0158   0.0077   0.0090
  0.0417   0.0408   0.0392   0.0367   0.0329   0.0277   0.0211   0.0135   0.0064   0.0150
  0.0416   0.0406   0.0387   0.0356   0.0314   0.0257   0.0189   0.0116   0.0128   0.0222
  0.0415   0.0403   0.0381   0.0346   0.0299   0.0239   0.0170   0.0191   0.0242   0.0314

 ••• best interface position
```

For each of the following cases, we numerically tested all possible interface positions and obtained the position that minimizes the error and the number of iterations to reach this accuracy. Following are the conclusions of our tests.

1. In the two-subdomain case without convection ($\beta = 0$), we obtain the best accuracy with a minimum overlap, $a - b = h_1$ and the ratio h_2/h_1 as well as $1 - b$ in agreement with the asymptotic estimates of Lemmas 1 and 2. The numerical error of the Schwarz alternating procedure reaches its minimum in a few (two or three) steps; see Table 1. Also, the numerical error is very sensitive to the position of the interface; Figure 1 shows the dependence of the error on the interface position when we assume a minimum overlap; the data are extracted from Table 1.

   The efficiency of the Schwarz procedure is impressive, and the method is certainly more efficient in terms of elapsed time for a given accuracy than a classical computation with one domain and a regular mesh.

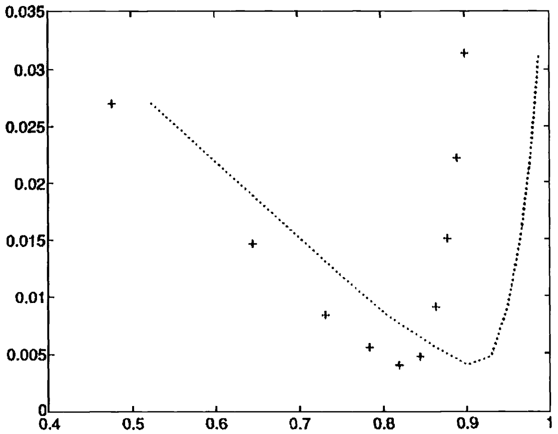


FIG. 1. *Dependence of the error on the interface position.* + *left interface of the boundary layer subdomain.* ··· *right interface of the outer subdomain.*

2. In the three-subdomain case without convection, one obtains the best accuracy when the intermediate subdomain almost completely overlaps the inner and outer subdomains. Therefore, the convergence rate of the method is still good; two or three iterations suffice, but the efficiency of the algorithm is less clear. We did not obtain an advantage over the previous two-subdomain strategy except that the inner and outer subdomain computations are decoupled. Numerically, the best position for the interfaces occurs when the accuracy in each subdomain is of the same order as it should be.

   We observed that one can get very good accuracy in the maximum norm on the composite grid when there is no representation of the solution in the layer—that is, when no discretization points are present in the layer. Hence, it is valuable to use any a priori knowledge of the existing layers given by the asymptotic analysis, and not only an adaptive scheme.
3. For the convective case ($\beta \neq 0$), we tested the Schwarz alternating procedure with a one-sided first-order difference scheme, as well as a centered second-order finite difference scheme. The adaptive domain decomposition renders the centered finite difference scheme very effective. The interface position is in agreement with the asymptotic analysis; see Tables 2 and 3.

   Again, because of the singular perturbation nature of the problem, the Schwarz alternating procedure converges in a few iterations, even with a minimum overlap ($b - a = h_1$) in the two-subdomain case, so the Schwarz method is still more efficient than a straightforward method with one domain and a regular mesh for the solution of a layer.

   Again, we observed numerically that the addition of more subdomains does not result in an improvement of the method.

3. Adaptive domain decomposition. One can have an accurate representation of the solution in the inner layer with three overlapping subdomains, two outer subdomains on each side of the inner layer, and one inner subdomain to solve the layer. For example, let $\Omega = [-1, 1]$ be the computational domain and S the location of an inner layer. By applying the previous asymptotic analysis for boundary layers on each of the subdomains $\Omega_l = [-1, S]$ and $\Omega_r = [S, 1]$, one obtains the best interface positions to solve the inner layer. In this section, we consider the following turning-point problem:

TABLE 2

Error in the maximum norm and number of iterations with two subdomains for the operator $-\varepsilon u'' + \beta u' + \gamma u$ and a decentered scheme (backward). $-\varepsilon u'' + \beta u' + \gamma u = f$ with $\beta = 1$, $\gamma = -0.5$, two subdomains with $N = 12$ and $\varepsilon = 0.02$ (one-sided scheme).

```
right boundary of the outer subdomain:

  0.0991   0.1089   0.1209   0.1358   0.1549   0.1803   0.2157   0.2683   0.3548   0.5238
  0.1964   0.2136   0.2340   0.2588   0.2895   0.3284   0.3793   0.4490   0.5500   0.7097
  0.2920   0.3143   0.3402   0.3708   0.4074   0.4521   0.5077   0.5789   0.6735   0.8049
  0.3860   0.4112   0.4400   0.4731   0.5116   0.5570   0.6111   0.6769   0.7586   0.8627
  0.4783   0.5046   0.5340   0.5670   0.6044   0.6471   0.6962   0.7534   0.8209   0.9016
  0.5690   0.5946   0.6226   0.6535   0.6875   0.7253   0.7674   0.8148   0.8684   0.9296 •••
  0.6581   0.6814   0.7064   0.7333   0.7624   0.7938   0.8280   0.8652   0.9059   0.9506
  0.7458   0.7652   0.7857   0.8073   0.8302   0.8544   0.8800   0.9072   0.9362   0.9670
  0.8319   0.8462   0.8609   0.8761   0.8919   0.9083   0.9252   0.9429   0.9612   0.9802
  0.9167   0.9244   0.9322   0.9402   0.9483   0.9565   0.9649   0.9735   0.9821   0.9910

left boundary of the boundary layer subdomain:

 0.0090   0.0198   0.0330   0.0494   0.0704   0.0984   0.1373   0.1951   0.2903   0.4762
 0.0179   0.0388   0.0638   0.0941   0.1316   0.1791   0.2414   0.3265   0.4500   0.6452
 0.0265   0.0571   0.0928   0.1348   0.1852   0.2466   c.3231   0.4211   0.5510   0.7317
 0.0351   0.0748   0.1200   0.1720   0.2326   0.3038   0.3889   0.4923   0.6207   0.7843
 0.0435   0.0917   0.1456   0.2062   0.2747   0.3529   0.4430   0.5479   0.6716   0.8197
 0.0517   0.1081   0.1698   0.2376   0.3125   0.3956   0.4884   0.5926   0.7105   0.8451 •••
 0.0598   0.1239   0.1927   0.2667   0.3465   0.4330   0.5269   0.6292   0.7412   0.8642
 0.0678   0.1391   0.2143   0.2936   0.3774   0.4660   0.5600   0.6598   0.7660   0.8791
 0.0756   0.1538   0.2348   0.3186   0.4054   0.4954   0.5888   0.6857   0.7864   0.8911
 0.0833   0.1681   0.2542   0.3419   0.4310   0.5217   0.6140   0.7080   0.8036   0.9009

number of iteration until convergence

   3    3    3    3    4    4    4    4    5    7
   2    3    3    3    3    3    3    4    4    6
   2    2    2    3    3    3    3    3    4    6
   2    2    2    2    2    3    3    3    4    6
   2    2    2    2    2    3    3    3    4    5
   2    2    2    2    2    2    3    3    4    5 •••
   2    2    2    2    2    2    3    3    4    5

   2    2    2    2    2    2    3    3    3    5
   2    2    2    2    2    2    3    3    3    5
   2    2    2    2    2    2    2    3    3    4

 error in the maximum norm:

  0.1421   0.1441   0.1465   0.1494   0.1531   0.1578   0.164    0.1722   0.1833   0.1906
  0.1437   0.1475   0.1518   0.1568   0.1627   0.1696   0.177    0.1856   0.1903   0.1676
  0.1451   0.1504   0.1562   0.1626   0.1697   0.1771   0.184    0.1891   0.1837   0.1379
  0.1464   0.1529   0.1599   0.1672   0.1747   0.1818   0.187    0.1872   0.1717   0.1129
  0.1476   0.1552   0.1629   0.1708   0.1782   0.1844   0.187    0.1821   0.1581   0.0941
  0.1487   0.1571   0.1655   0.1735   0.1806   0.1853   0.185    0.1752   0.1447   0.0832 •••
  0.1496   0.1588   0.1676   0.1756   0.1819   0.1851   0.182    0.1675   0.1325   0.0848
  0.1505   0.1602   0.1692   0.1770   0.1825   0.1839   0.178    0.1595   0.1220   0.0909
  0.1512   0.1614   0.1705   0.1780   0.1825   0.1820   0.173    0.1517   0.1139   0.1051
  0.1518   0.1624   0.1716   0.1785   0.1819   0.1796   0.168    0.1447   0.1089   0.1284

••• best interface position
```

$$\varepsilon \frac{d^2}{dx^2} u + x^{\gamma} \frac{d}{dx} u = f, \quad u(-1) = 1, \quad u(1) = -1,$$

where γ is an odd integer. This turning-point problem has an inner layer at $S = 0$, and we alternate the computation on the outer domains $\Omega_1 = [-1, -b]$ and $\Omega_3 = [b, 1]$ with the computation on the overlapping subdomain $\Omega_2 = [-a, a]$. The thickness of the transition layer is $\varepsilon^{1/(\gamma+1)}$, and there exists a corrector that vanishes exponentially outside the layer. Therefore, we must choose $b \approx \varepsilon^{1/(\gamma+1)} \log(\varepsilon^{-1})$. Since the coefficient of the convection term is small in the overlap area,

$$\beta(x) = x^{\gamma} \approx \varepsilon^{\gamma/(\gamma+1)} \log(\varepsilon) \text{ on } (b, a) \text{ and } (-a, -b),$$

TABLE 3

Error in the maximum norm and number of iterations with two subdomains for the operator $-\varepsilon u'' + \beta u' + \gamma u$ *and a centered scheme.* $-\varepsilon u'' + \beta u' + \gamma u = f$ *with* $\beta = 1$, $\gamma = -0.5$, *two subdomains with* $N = 12$ *and* $\varepsilon = 0.02$ *(centered scheme).*

```
rignt boundary of the outer subdomain:

    0.0991   0.1089   0.1209   0.1358   0.1549   0.1803   0.2157   0.2683   0.3548   0.5238
    0.1964   0.2136   0.2340   0.2588   0.2895   0.3284   0.3793   0.4490   0.5500   0.7097
    0.2920   0.3143   0.3402   0.3708   0.4074   0.4521   0.5077   0.5789   0.6735   0.8049
    0.3860   0.4112   0.4400   0.4731   0.5116   0.5570   0.6111   0.6769   0.7586   0.8627
    0.4783   0.5046   0.5340   0.5670   0.6044   0.6471   0.6962   0.7534   0.8209   0.9016
    0.5690   0.5946   0.6226   0.6535   0.6875   0.7253   0.7674   0.8148   0.8684   0.9296 •••
    0.6581   0.6814   0.7064   0.7333   0.7624   0.7938   0.8280   0.8652   0.9059   0.9506
    0.7458   0.7652   0.7857   0.8073   0.8302   0.8544   0.8800   0.9072   0.9362   0.9670
    0.8319   0.8462   0.8609   0.8761   0.8919   0.9083   0.9252   0.9429   0.9612   0.9802
    0.9167   0.9244   0.9322   0.9402   0.9483   0.9565   0.9649   0.9735   0.9821   0.9910


left boundary of the boundary layer subdomain:

    0.0090   0.0198   0.0330   0.0494   0.0704   0.0984   0.1373   0.1951   0.2903   0.4762
    0.0179   0.0388   0.0638   0.0941   0.1316   0.1791   0.2414   0.3265   0.4500   0.6452
    0.0265   0.0571   0.0928   0.1348   0.1852   0.2466   0.3231   0.4211   0.5510   0.7317
    0.0351   0.0748   0.1200   0.1720   0.2326   0.3038   0.3889   0.4923   0.6207   0.7843
    0.0435   0.0917   0.1456   0.2062   0.2747   0.3529   0.4430   0.5479   0.6716   0.8197
    0.0517   0.1081   0.1698   0.2376   0.3125   0.3956   0.4884   0.5926   0.7105   0.8451 •••
    0.0598   0.1239   0.1927   0.2667   0.3465   0.4330   0.5269   0.6292   0.7412   0.8642
    0.0678   0.1391   0.2143   0.2936   0.3774   0.4660   0.5600   0.6598   0.7660   0.8791
    0.0756   0.1538   0.2348   0.3186   0.4054   0.4954   0.5888   0.6857   0.7864   0.8911
    0.0833   0.1681   0.2542   0.3419   0.4310   0.5217   0.6140   0.7080   0.8036   0.9009


number of iteration until convergence

    3     3     3     3     3     3     3     3     3     4
    2     2     2     2     2     2     2     2     3     6
    2     2     2     2     2     2     2     3     4     7
    2     2     2     2     2     2     2     3     4     8
    2     2     2     2     2     2     3     3     4     8
    2     2     2     2     2     2     3     3     5     8 •••
    2     2     2     2     2     3     3     4     5     8
    2     2     2     2     2     3     3     4     5     8
    2     2     2     2     3     3     3     4     5     7

    2     2     2     2     3     3     3     4     5     5

 error in the maximum norm:

    0.4154   0.4109   0.4053   0.3982   0.3891   0.3767   0.35   0.3314   0.2838   0.1841
    0.4117   0.4028   0.3920   0.3786   0.3615   0.3391   0.30   0.2649   0.1984   0.0941
    0.4080   0.3949   0.3792   0.3600   0.3362   0.3060   0.26   0.2142   0.1432   0.0537
    0.4044   0.3872   0.3668   0.3425   0.3131   0.2768   0.23   0.1752   0.1063   0.0339
    0.4008   0.3796   0.3550   0.3261   0.2918   0.2510   0.20   0.1449   0.0810   0.0235
    0.3972   0.3723   0.3436   0.3105   0.2723   0.2281   0.17   0.1210   0.0630   0.0217 •••
    0.3937   0.3651   0.3326   0.2959   0.2544   0.2078   0.15   0.1020   0.0494   0.0310
    0.3902   0.3580   0.3221   0.2821   0.2380   0.1897   0.13   0.0869   0.0366   0.0603
    0.3868   0.3512   0.3120   0.2692   0.2227   0.1736   0.12   0.0758   0.0581   0.1212
    0.3835   0.3444   0.3027   0.2566   0.2095   0.1588   0.10   0.0807   0.1780   0.2293

••• best interface position
```

the convergence rate of the Schwarz alternating procedure defined below is better than stated in Lemmas 5 and 6.

We use a one-sided first-order finite difference approximation of $x^{\gamma} du/dx$, oriented depending on the sign of x. For the interval [0, 1] we obtain the following result. Let $\delta \succ\succ 1$ and $N \sim \varepsilon^{-1/(\gamma+1)}/\delta$. Then

$$\zeta_{\leftarrow} \sim 1, \quad \zeta_{\rightarrow} \sim (C\delta \log(\varepsilon^{-1}))^{-k_0}.$$

A similar result holds for the interval $[-1, 0]$.

Our numerical experiments confirm that we get the best numerical accuracy with a minimum overlap, $b - a = h_1 = h_3$, a ratio h_1/h_2 in agreement with the asymptotic analysis, and few iterations. In our experiments, we used various values of γ to vary the thickness of the layer.

The main problem with inner layers is that the asymptotic analysis gives only a rough estimate of their location. We propose an adaptive Schwarz procedure to refine the location of the layers based on the following procedure.

We start with a given domain decomposition based on a rough localization of the layer. During the iteration process of the Schwarz procedure, we let the interfaces of the subdomains evolve according to the equilibrium principle of the numerical error in each subdomain. This criterion has been shown in Lemmas 1–3 to give the best interface position. We measure the error in each subdomain based on a numerical approximation of

$$E_1 = Ch_1^n \max\left(\frac{d^{n_1+p_1}}{dx^{n_1+p_1}}u\right) \text{ for } \Omega_1 = [-1, b_-],$$

$$E_3 = Ch_3^n \max\left(\frac{d^{n_3+p_3}}{dx^{n_3+p_3}}u\right) \text{ for } \Omega_3 = [b_+, 0],$$

$$E_2 = C(a_+ - a_-)^{q_2} h_2^n \max\left(\frac{d^{n_2+p_2}}{dx^{n_2+p_2}}u\right) \text{ for } \Omega_2 = [a_-, a_+],$$

where p_i, n_i, and q_i depend on the discretization of the operator in the corresponding subdomain. It is convenient to use the fact that the numerical error in the outer domains is dominated by the error at the endpoint $b_\pm$.

We have implemented this adaptive procedure on this turning-point problem with $\gamma = 1$. We still restrict ourselves to letting the subgrids coincide at their endpoints and need to interpolate the solution on the new subgrids for each displacement of the interfaces. We use a decentered first-order finite difference scheme in the outer domains and a second-order centered difference scheme in the inner subdomain. Figure 2e shows the evolution of the interfaces during the adaptivity process. The initial position of the interfaces gives the solution plotted in Figure 2a; then our adaptivity criterion gives the sequence of the solutions plotted in Figure 2b, 2c, 2d, and, finally, 2c. For the solution plotted in Figure 2c, the numerical error in each subdomain balances. We observe that the limit of the interface position is in agreement with the asymptotic analysis.

Our measure of the error in each subdomain is relatively rough and a further step in this research is to use a posteriori local estimates to drive the adaptivity of the interface positions.

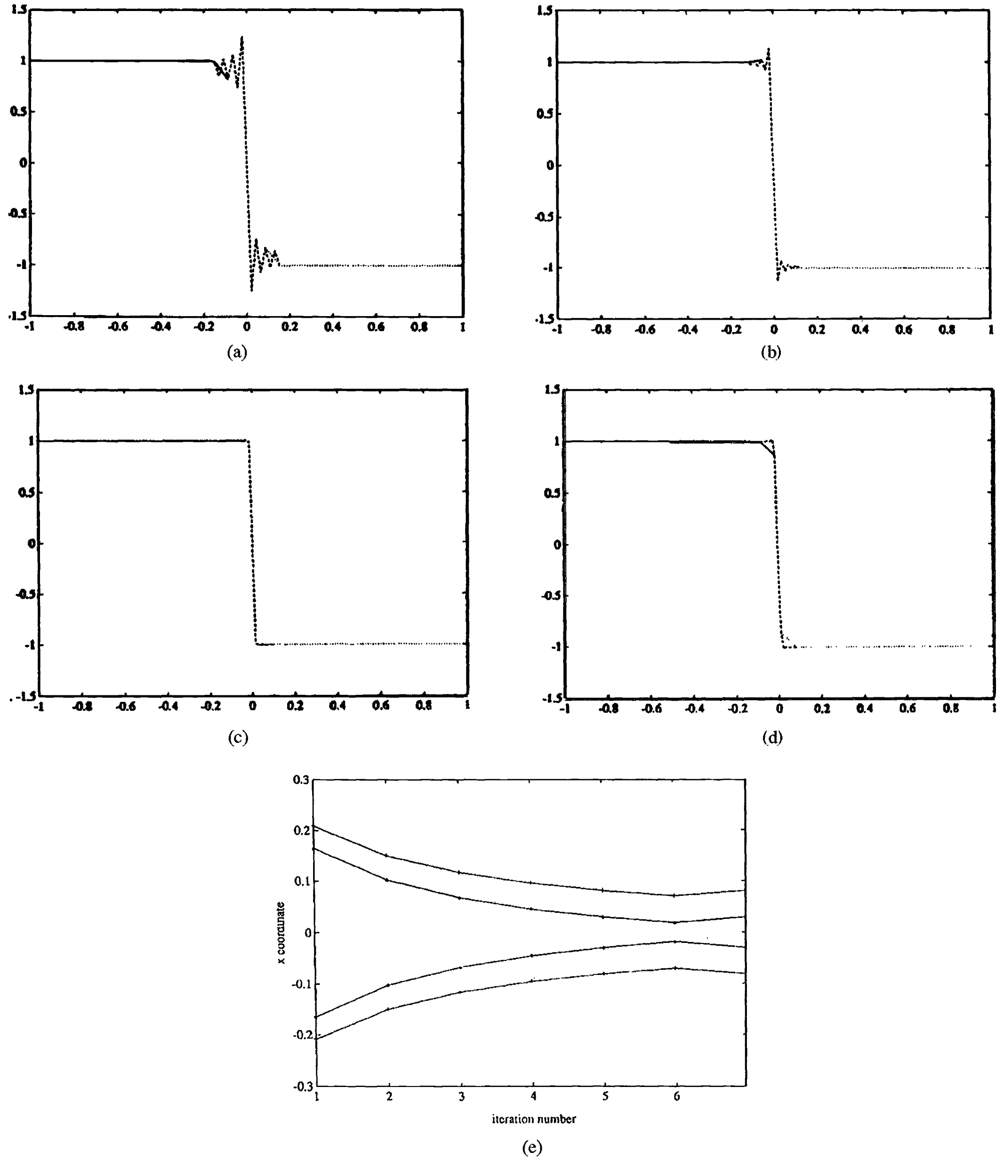
Remark 5. Suppose that we use a finite difference scheme that satisfies a maximum principle inside each subdomain. We observe that the evolution of the interfaces does not necessarily slow down the speed of convergence of the Schwarz procedure when, for example, a subdomain Ω_i shrinks *and* when the measure of its overlap with the subdomains Ω_{i-1} and Ω_{i+1} stays the same.

4. ADI and domain decomposition for the two-dimensional problem. We consider the following two-dimensional turning-point problem, P_ε:

$$L[u] = \varepsilon^2 \Delta u + a(x, y)\frac{\partial u}{\partial x} = 0 \text{ in } \Omega = (-2, 2) \times (-1, 1),$$

$$u(-2, y) = 1, u(2, y) = -1, \ y \in [-1, 1], \tag{4.25}$$

$$\frac{\partial u}{\partial y}(x, \pm 1) = 0, \ x \in [-2, 2],$$



(a)

(b)

(c)

(d)

(e)

FIG. 2. *Turning-point problem in one space dimension.*

where

$$a(x, y) = \begin{cases} x & \text{for } y < 0\,, \\ x - y & \text{for } y > 0\,. \end{cases}$$

Let $S(y)$ be the curve parameterized by y, such that

$$x = S(y) \Leftrightarrow a(x, y) = 0.$$

An asymptotic analysis shows that $u = 1 + o(1)$ $(u = -1 + o(1))$ to the left (right) of a transition layer centered at S. Also, the thickness of this layer is of the order of ε.

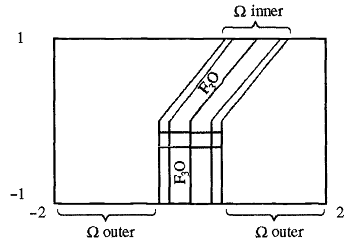


FIG. 3. *Turning-point problem in two space dimensions.*

We have tested the Schwarz alternating procedure on this turning-point problem in two space dimensions. The numerical challenge is to solve an inner layer centered on a curve that is not smooth. Inner layers are characteristic of so-called frontal processes; examples are combustion fronts, polymerization fronts, and reactive shock layers. These problems are truly nonlinear, and it is well known that even a linear turning-point problem is numerically difficult to solve [31].

Let Ω^h be a regular two-dimensional grid of mesh size h. We first solve P_{ε_0} approximately on the grid Ω^h with $\varepsilon = \varepsilon_0$, such that $1 \succ\succ \varepsilon_0 \succ\succ h$. Our goal is to solve P_ε with $\varepsilon \prec\prec h$. We then use this solution as an initial guess for the Schwarz alternating procedure.

To discretize the operator, we use a second-order centered finite difference approximation of the second-order derivatives and a one-sided finite difference approximation of $a(x, y)\partial u/\partial x$ that depends on the sign of $a(x, y)$ as usual. We choose a very simple discretization, as we only want to test the Schwarz alternating procedure. To have an accurate representation of the transition layer, we introduce the following domain decomposition:

$$\Omega^{\text{outer}}_{\text{west}} = \{M(x, y) \in \Omega,\ |x - S(y)| \geq (d-1)h, x - S(y) < 0\},$$

$$\Omega^{\text{outer}}_{\text{east}} = \{M(x, y) \in \Omega,\ |x - S(y)| \geq (d-1)h, x - S(y) > 0\},$$

$$\Omega^{\text{inner}}_{\text{north}} = \{M(x, y) \in \Omega,\ |x - S(y)| \leq dh, y > -dh\},$$

$$\Omega^{\text{inner}}_{\text{south}} = \{M(x, y) \in \Omega,\ |x - S(y)| \leq dh, y < -(d-1)h\},$$

where d is a fixed positive integer. To demonstrate the efficiency of the Schwarz alternating procedure in the worst possible case, we assume a minimum overlap between the subdomains; see Figure 3.

A standard asymptotic argument shows that we have a two-dimensional layer in a neighborhood Ω^* of the corner point (0, 0) of S. One needs to stretch *both* variables x and y to solve this internal layer; $\hat{x} = x/\varepsilon$ and $\hat{y} = y/\varepsilon$ are the inner variables. Outside Ω^*, the transition layer is really a one-dimensional phenomenon, and one needs to stretch only the layer in the direction normal to the front. For example, in $\Omega^{\text{inner}}_{\text{south}}$, the inner variables are $(\hat{x}, y)$.

Globally, the transition layer is of thickness ε in the x-direction for $y \in (-1, 1)$, and there exists a corrector to the transition layer that vanishes exponentially outside the layer, so we take $dh = O(\varepsilon \log \varepsilon^{-1})$.

To simplify the implementation, we extended the subdomain that solves the inner layer in the neighborhood of (0, 0) to $\Omega^{\text{inner}}_{\text{north}}$ with the same stretching for both variables.

The discretization grid for the inner problems are refinements of the global rough grid. In $\Omega^{\text{inner}}_{\text{south}}$, we use a regular grid of step $\hat{h} = h\varepsilon$ in the x-direction and h in the y-direction. In $\Omega^{\text{inner}}_{\text{north}}$, we use a regular grid of step $\hat{h} = h\varepsilon$ in both directions x and y.

We are going to use a strip domain decomposition at each stage of the algorithm in order to keep the same convergence properties as in the one-dimensional case. The numerical algorithm is the following (we do not repeat the boundary conditions on $\partial\Omega$):

While $\|u^{p+1} - u^p\|_\Omega \prec \text{tol}$,

- solve the outer problems (eventually in parallel),

$$L[u_{\text{outer}}^{\text{west},p+1}] = 0 \text{ in } \Omega_{\text{outer}}^{\text{west}},$$

$$u_{\text{outer}}^{\text{west},p+1} = u_{\text{inner}}^{p} \text{ on } \Gamma_{\text{west}} = \{(S(y) - (d-1)h, y), y \in [-1,1]\},$$

$$L[u_{\text{outer}}^{\text{east},p+1}] = 0 \text{ in } \Omega_{\text{outer}}^{\text{east}},$$

$$u_{\text{outer}}^{\text{east},p+1} = u_{\text{inner}}^{p} \text{ on } \Gamma_{\text{east}} = \{(S(y) - (d-1)h, y), y \in [-1,1]\};$$

- solve the inner problem (P_{inner}),

$$L[u_{\text{inner}}^{p+1}] = 0 \text{ in } \Omega_{\text{inner}} = \Omega_{\text{south}}^{\text{inner}} \cup \Omega_{\text{north}}^{\text{inner}},$$

$$u_{\text{inner}}^{p+1} = u_{\text{outer}}^{\text{west},p+1} \text{ on } \Gamma_{\text{west}}^{*} = \{(S(y) - dh, y), y \in [-1,1]\},$$

$$u_{\text{inner}}^{p+1} = u_{\text{outer}}^{\text{est},p+1} \text{ on } \Gamma_{\text{est}}^{*} = \{(S(y) + dh, y), y \in [-1,1]\};$$

- define u^{p+1} by

$$u^{p+1} = \begin{cases} u_{\text{outer}}^{p+1} & \text{in } \Omega_{\text{outer}} \backslash \Omega_{\text{inner}}, \\ u_{\text{inner}}^{p+1} & \text{in } \Omega_{\text{inner}}. \end{cases}$$

end while.

To solve the inner problem (P_{inner}), we use a second Schwarz alternating procedure inside the previous loop.

While $\|u_{\text{inner}}^{p,q+1} - u_{\text{inner}}^{p,q}\| \leq \text{tol}$,

- solve the south inner problem,

$$L[u_{\text{inner}}^{\text{south},p,q+1}] = 0 \text{ in } \Omega_{\text{south}}^{\text{inner}},$$

$$u_{\text{inner}}^{\text{south},p,q+1} = u_{\text{inner}}^{\text{north},p,q} \text{ on } \Gamma = \{(x, -(d-1)h), x \in [-dh, dh]\};$$

- solve the north inner problem,

$$L[u_{\text{inner}}^{\text{north},p,q+1}] = 0 \text{ in } \Omega_{\text{north}}^{\text{inner}},$$

$$u_{\text{inner}}^{\text{north},p,q+1} = u_{\text{inner}}^{\text{south},p,q+1} \text{ on } \Gamma = \{(x, -dh), x \in [-dh, dh]\};$$

- define $u_{\text{inner}}^{p,q+1}$ by

$$u_{\text{inner}}^{p,q+1} = \begin{cases} u_{\text{inner}}^{\text{south},p,q+1} & \text{in } \Omega_{\text{south}}^{\text{inner}} \backslash \Omega_{\text{north}}^{\text{inner}}, \\ u_{\text{inner}}^{\text{north},p,q+1} & \text{in } \Omega_{\text{north}}^{\text{inner}}. \end{cases}$$

end while.

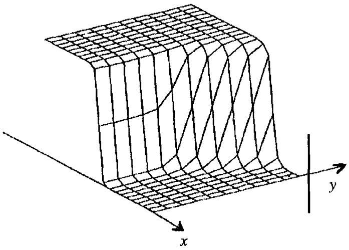


FIG. 4. *Domain decomposition.* $\varepsilon^2 \Delta u + F(x, y) u_x = 0$. *Solution projected on the rough grid.*

Let u^p_{inner} be the result of the last computation of this second Schwarz alternating procedure. The initial guess for $u^{p,0}_{\text{inner}}$ is obviously u^{p-1}_{inner}. Therefore, as the first Schwarz alternating procedure proceeds, the inner loop of the second Schwarz alternating procedure takes fewer and fewer iterations to converge.

To complete the description of this algorithm, we give the following details. We solved each subproblem with an ADI method [10]. This method has at least four advantages:

- It is a nice way to get an efficient massively parallel algorithm. We really need a powerful computer, as frontal processes are computationally very intensive.
- It induces a weak coupling between the computations in each subdomain because it is an explicit scheme with respect to one of the space directions.
- It is an easy way to solve each subproblem with a nontrivial geometry. This is important because we need to adapt the shape of the inner domain according to the location of the layer.
- It is an iterative procedure. One can start the ADI iteration with the last-known subdomain solution. As the Schwarz iteration progresses, the number of ADI iterations required decreases. Also, we do not need to solve each subproblem very accurately at the beginning of the Schwarz iterations.

We observe that our domain decomposition and stretching in the inner subdomains make each subproblem nonstiff and suitable for an ADI solver.

Our numerical experiments were obtained with $\varepsilon = 10^{-1}$, $d = 2$, and $h = 5 \times 10^{-2}$. Thus, the refinement of the grid is $\hat{h} = h/10$; see Figures 4 and 5. The number of iterations for the loop that alternates the computation in the north and south inner domains was held fixed at two. The evolution of $\|u^{p+1} - u^p\|_\Omega$ as p increases shows that, above $p = 4$, $\|u^{p+1} - u^p\|_\Omega$ is dominated by the discretization error; see Figure 6. Therefore, this experiment confirms that the Schwarz alternating procedure converges in a few iterations, as in the one-dimensional case.

We intend to implement these ideas for arbitrarily shaped curves S and local coordinate systems that stretch only the *normal component* of S.

5. Nonlinear combustion problem. So far, we have shown the efficiency of the Schwarz alternating procedure for linear second-order singular perturbation problems whose formal limit as $\varepsilon \rightarrow 0$ is a zero- or first-order operator. Now, we illustrate the domain decomposition method for a nonlinear singular perturbation problem arising in combustion [2]. The problem involves two variables— θ, a scaled temperature, and C, a concentration of reactant, both functions of the polar coordinates r and ψ— and corresponds to a thermodiffusive model for a one-step Arrhenius reaction [25]. The equations satisfied by θ and C are

$$\theta_t = \theta_{rr} + \frac{1}{r^2}\theta_{\psi\psi} + (1 - K)\frac{\theta_r}{r} + A_r(\theta, C),$$

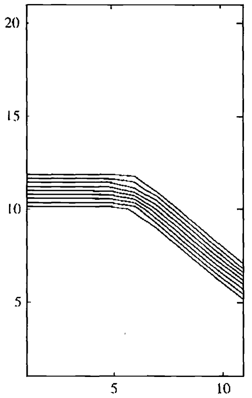


FIG. 5 *Solution in the inner layer.*

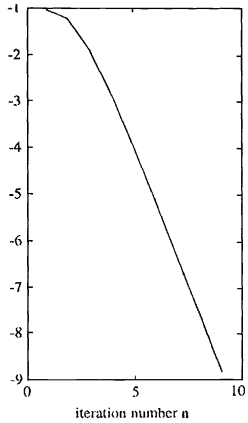


FIG. 6 *Convergence history.* $\log(u(n+1)-u(n))$.

$$C_t = \frac{1}{L}\left(C_{rr} + \frac{1}{r^2}C_{\psi\psi}\right) + \left(\frac{1}{L} - K\right)\frac{Cr}{r} - A_r(\theta, C),$$

where A_r is a nonlinear source term

$$A_r(\theta, C) = \beta C \exp\frac{N(1-\sigma)(\theta-1)}{\sigma+(1-\sigma)\theta}.$$

The parameters are L, the Lewis number; N, the (scaled) activation energy; and K, a measure for the strength of the fuel injection; furthermore, $\beta = M^2/2L$ and $M = N(1-\sigma)$. The parameters K and N are both supposed to be large.

Both θ and C are periodic functions of ψ. The boundary conditions satisfied by θ and C are

$$\theta \to 0,\ C \to 1 \text{ as } r \to 0; \quad \theta \to 1,\ C \to 0 \text{ as } r \to \infty.$$

For the computational domain we take $(r, \psi) \in (r_0, r_1) \times (0, 2\pi)$, where $0 < r_0 < r_1 < \infty$; r_0 is sufficiently small and r_1 is sufficiently large so that the boundary conditions can be replaced by the Dirichlet conditions

$$\theta(r_0, \psi) = 0,\ C(r_0, \psi) = 1; \quad \theta(r_1, \psi) = 1,\ C(r_1, \psi) = 0.$$

When $N \gg 1$, the flame is a thin layer, and the combustion problem is of singular perturbation type. The asymptotic analysis can be found in [25]. When $N \to \infty$ (infinite activation energy) and $L = 1$ (equal coefficients for mass and heat diffusion), the basic solution is given by

$$(5.26)\quad \theta(r,\psi) = (r/K)^K \text{ for } r \le K,\ \theta(r,\psi) = 1 \text{ for } r \ge K, \quad C(r,\psi) = 1 - \theta(r,\psi).$$

This solution represents a cylindrical combustion front.

For finite activation energies, θ and C are smooth functions which exhibit a transition layer of thickness $0(1/N)$. The transition layer is centered at $r = K$. The nonlinear source term $A_r(\theta, C)$ vanishes exponentially outside the layer. When $L < 1$, the cylindrical combustion front becomes unstable, and a cellular flame appears [25]. Moreover, the location of the front is a perturbation of K.

To compute the combustion front accurately, we use a three-subdomain strategy. We assume that the combustion front corresponds to the curve $r = S(\psi)$, $\psi \in (0, 2\pi)$, such that $A_r(S(\psi), \psi) = \max_{r\in(r_0,r_1)} A_r(r, \psi)$.

A Hopf bifurcation occurs when the Lewis number L reaches a critical value $L_0 < 1$. When L is not too far from L_0, the amplitude of the cells of the combustion front is small compared with the size of the domain, so we assume

$$\max_{\psi} S - \min_{\psi} S \ll r_1 - r_0$$

and use a strip domain decomposition technique. Let Ω_l^{outer}, Ω^{inner}, and Ω_r^{outer} be the subdomains. We impose a minimum overlap between the outer and inner domains. We notice that the Schwarz alternating procedure applied to the time-dependent problem converges rather well for small time steps, but we want to illustrate two other possible advantages of the domain decomposition to solve the combustion front.

5.1. Large activation energy. Because $N \gg 1$, the source term A_r vanishes exponentially outside the neighborhood of the combustion front. In fact, the location of the transition layer corresponds to the zone where the source term is strictly of order one. Therefore, the right boundary P of Ω_l^{outer} and the left boundary G of Ω_r^{outer} can be chosen *adaptively*, based on the asymptotic criterion

$$\max_{\Omega_l, \Omega_r} A_r < \text{tol}, \tag{5.27}$$

where tol is a very small tolerance number fixed in advance.

We use the following semi-implicit scheme in the left outer domain:

$$L[\theta^{n+1}] = \theta^n + \Delta t \ A_r(\theta^n, C^n) \text{ in } \Omega_{\text{left}}^{\text{outer}}, \tag{5.28}$$

$$L[C^{n+1}] = C^n - \Delta t \ A_r(\theta^n, C^n) \text{ in } \Omega_{\text{left}}^{\text{outer}}, \tag{5.29}$$

$$\theta^{n+1}(r_0, \psi) = 0, \quad C^{n+1}(r_0, \psi) = 1, \qquad \psi \in (0, 2\pi),$$

$$\theta^{n+1}(P, \psi) = \theta^{\text{inner}}(P, \psi), \qquad C^{n+1}(P, \psi) = C^{\text{inner}}(P, \psi).$$

Here, L is the operator

$$L[\theta] = \theta - \Delta t \left(\theta_{rr} + \frac{1}{r^2}\theta_{\psi,\psi} + (1 - K)\frac{\theta r}{r} \right),$$

and θ^{n+1} and C^{n+1} are periodic in ψ. The values θ^{inner} and C^{inner} are given artificial boundary conditions.

Since L satisfies a maximum principle, we can estimate the influence of the source term A_r,

$$\|\theta^{n+1} - \theta_0\|_\infty \le C\Delta t \|A_r\|_\infty, \qquad \|C^{n+1} - C_0\|_\infty \le C\Delta t \|A_r\|_\infty,$$

where (θ_0, C_0) is the solution when $A_r = 0$. Hence, if tol in (5.27) is small enough, the explicit treatment of the nonlinear source terms in the outer domains is justified [15]. In fact, we may choose not to compute these source terms at every time step, or simply neglect them altogether in the outer domains. In any case, the equations for the temperature (5.28) and for the concentration (5.29) can be solved in parallel.

It is well known that the computation of the flame front is very sensitive to the accuracy of the scheme. Since we have a strong interaction between the diffusion process and the nonlinear reaction term A_r *only* in the neighborhood of the combustion front, we need to use a highly accurate scheme in the transition layer; on the contrary, a less accurate scheme may be good enough in the outer domains where $\max_{\Omega_l,\Omega_r} A_r < \text{tol}$.

5.2. Strong fuel injection. As $K \gg 1$, r is large in the neighborhood of the combustion front. When the combustion front $r = S(\psi)$ is a smooth curve, we have

$$\frac{1}{r^2}\theta_{\psi\psi} \prec\prec \theta_{rr} \text{ on } \Omega^{\text{inner}}. \tag{5.30}$$

This asymptotic behavior makes the use of an explicit scheme with respect to the angle dependence very attractive. We therefore consider the following scheme:

$$\hat{L}[\theta^{n+1}] = \theta^n + \Delta t\left(\frac{1}{r^2}\theta^n_{\psi\psi} + A_r(\theta^n, C^n)\right) \text{ on } \Omega^{\text{inner}}, \tag{5.31}$$

$$\hat{L}[C^{n+1}] = C^n + \Delta t\left(\frac{1}{r^2}C^n_{\psi\psi} - A_r(\theta^n, C^n)\right) \text{ on } \Omega^{\text{inner}}. \tag{5.32}$$

The penalty on the time step is moderated by the relation (5.30). Also, equations (5.31) and (5.32) represent a parallel scheme for the most intensive part of the computation, that is, in the transition layer. One can map a ring of processors to the set of equations (5.31) and (5.32) for $\psi \in \{\psi_i = 2\pi i/n : i = 0, \dots, n\}$ [8].

We have implemented the Schwarz alternating procedure with moving interfaces to solve a steady flame with the parameter values $N = 20$, $\sigma = 0.615$, $K = 14.8$. The domain of computation was $(1, 41) \times (0, 2\pi)$. The ratio of the space step for the radial direction in the outer and inner domain was N, equal to the asymptotic thickness of the layer.

Figure 7 shows the solution $(\theta, C, A_r(\theta, C))$ and the convergence history for decreasing values of the Lewis number L in the one-dimensional case. The combustion front moves slowly on the left as the Lewis number goes from $L = 1$ to $L = 0.7$. We used an iteration analogous to (5.31) and (5.32) to solve each subproblem of the Schwarz alternating procedure but with a domain-dependent relaxation factor. The nonlinear terms were solved explicitly in the scheme, but since most of the convergence process occurs in the transition layer, it might be valuable to use a Newton-like scheme in the inner subdomain to take a larger time step.

The convergence of the Schwarz iteration is very sensitive to the position of the left boundary of the inner domain and is improving as the left boundary moves on the left; see Figure 8. This can be explained in the following way: as the left boundary of the inner domain moves on the left, the convection factor $(K - 1)/r$ is getting larger in the overlap zone with the left outer domain and the operator becomes convection dominant. This is in agreement with the result of §2.2.1 and can be proved, for example, with the linearized operator for this domain decomposition. We recall that we keep the overlap to be only one cell of the rough grid.

Finally, when we cut off the source term $A_r(\theta, C)$ in the outer domains, this numerical experiment shows that the error is less than the numerical accuracy of the discretization with this domain decomposition.

In the two-dimensional case, we simply incorporated the explicit dependence of the second-order derivative with respect to the angle ψ in the scheme and used a fourth-order finite difference scheme for $\theta_{\psi\psi}$ and $C_{\psi\psi}$ in the transition layer. For convenience and in order to compute a more interesting case, we have computed a cellular flame for a *two step chemistry*. We consider the model of Pelaez and Linan [28]. We have compared our result to the numerical result using adaptive pseudospectral method for the case where the first Lewis number is $L_1 = 0.37$ and the second Lewis number is $L_2 = 0.9$. The heat release of the first reaction is 0.85. So the second chemical reaction gives a rather weak front, and the stiff front is mainly due to the first chemical reaction. For both reactions the activation energy is equal to 20. Figure 9 shows a representation of this cellular flame. Since the front is stiff mainly in the radial direction, we used the same number of discretization points, $\{\psi_i : i = 1, \dots, n\}$,

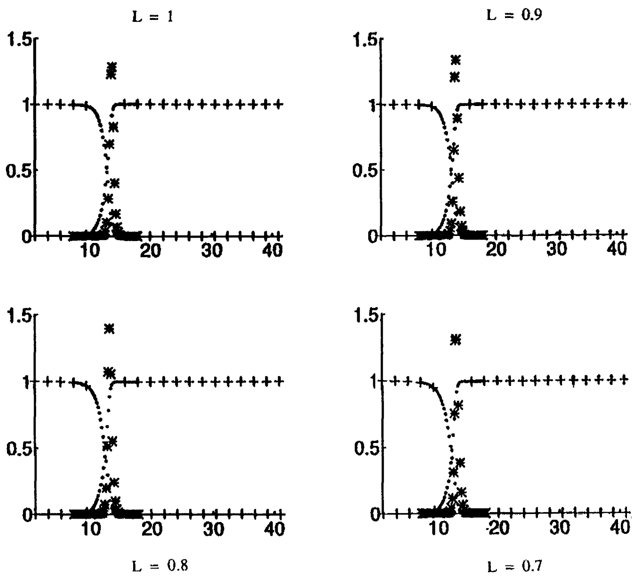


FIG. 7. *Combustion problem in one space dimension for different values of the Lewis number. * temperature, concentration, and Arrhenius term.*

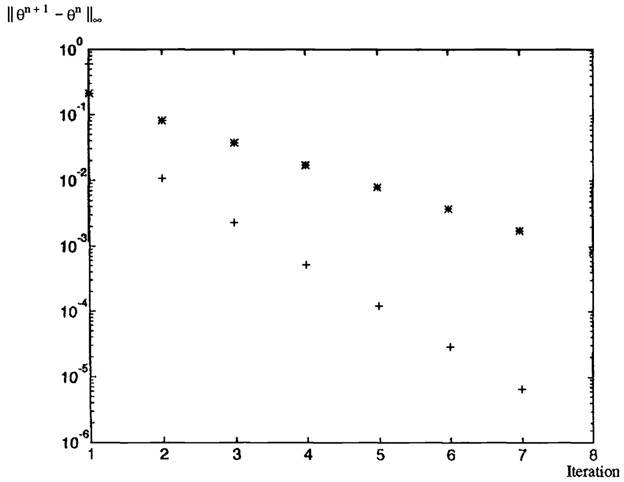


FIG. 8. *Combustion problem in one space dimension for different values of the Lewis number. * left interface of Ω inner at $x = 7.3$. + left interface of Ω inner at $x = 5.2$.*

for each subdomain with $n = 64$. We have used 20 discretization points in the radial direction for the rough grid, and the ratio of the space step for the radial direction in the outer and inner domain is equal to the activation energy.

We found good global agreement between the Schwarz procedure and the pseudospectral method of [2] but did not compute a very difficult case with $L_1 < 0.37$. More precisely, we have compared an adaptive domain decomposition with the pseudospectral method using 80 Chebyshev collocation points in the radial directions and 64 Fourier discretization points in the angular direction with the Schwarz procedure described above. We believe that the pseudospectral method is more accurate and we observe that the difference between the two solutions is of order 10^{-2} on the rough grid in the maximum norm.

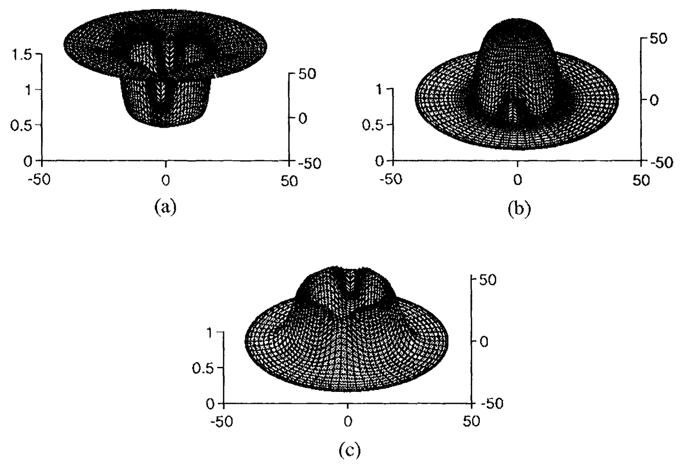


FIG. 9. *A cellular flame with two-step Arrhenius kinetics.* (*a*) *Temperature* ($L1 = 0.37$; $L2 = 0.90$). (*b*) *Concentration of the first species* ($L1 = 0.37$; $L2 = 0.90$). (*c*) *Concentration of the second species* ($L1 = 0.37$; $L2 = 0.90$).

Since the cost of the computation with the Schwarz alternating procedure is very cheap compared with the cost of computation of the adaptive pseudospectral domain decomposition, we expect that this new technique may be helpful to compute stationary cellular flame in three space dimensions. Also, an efficient implementation of this algorithm on a massively parallel computer is easy.

Acknowledgments. The author is very grateful to H. G. Kaper for many interesting discussions and the referee for some valuable comments on this paper.

REFERENCES

[1] J. BARANGER, *On the thickness of the boundary layer in elliptic singular perturbation problems*, in Numerical Analysis of Singular Perturbation Problems, P. W. Hemker and J. J. H. Miller, eds., Academic Press, London, 1979, pp. 395–400.

[2] A. BAYLISS, B. J. MATKOWSKY, AND M. MINKOFF, *Adaptive pseudospectral computation of cellular flame stabilized by a point source*, Appl. Math. Lett., 1 (1987), pp. 19–24.

[3] N. BOURBAKI, *Fonctions d'une Variable Réelle*, Hermann, Paris, 1976.

[4] A. BOURGEAT AND M. GARBEY, *Computation of viscous or nonviscous conservation laws by domain decomposition based on asymptotic analysis*, Numer. Methods Partial Differential Equations, 8 (1992), pp. 127–142.

[5] ZHIQIANG CAI AND S. MCCORMICK, *Computational complexity of the Schwarz alternating procedure*, Internat. J. High-Speed Comput., 1 (1989), pp. 1–28.

[6] R. C. Y. CHIN, G. W. HEDSTROM, J. R. GRAW, AND F. A. HOWES, *Parallel computation of multiple scale problems*, in New Computing Environments: Parallel, Vector and Systolic, A. Wouk, ed., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1986, pp. 136–153.

[7] R. C. Y. CHIN AND T. MANTEUFFEL, *An analysis of block successive overrelaxation for a class of matrices with complex spectra*, SIAM J. Numer. Anal., 25 (1988), pp. 564–585.

[8] F. DESPREZ AND M. GARBEY, *Numerical simulation of a combustion problem on a paragon machine*, Parallel Comput., 21 (1995), pp. 495–508.

[9] W. ECKHAUS, *Asymptotic analysis of singular perturbations*, Stud. Math. Appl., 9 (1979).

[10] D. J. EVANS, *The parallel AGE method for the elliptic problem in two dimensions*, Parallel Comput., 17 (1991), pp. 925–940.

[11] M. GARBEY, *Domain decomposition to solve singular perturbations and asymptotic*, in Fifth International Symposium on Domain Decomposition for Partial Differential Equations, D. E. Keyes et al., eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1991, pp. 293–305.

[12] ———, *Domain decomposition to solve layers and asymptotic*, SIAM J. Sci. Comput., 15 (1994), pp. 866–891.

[13] M. Garbey and H. G. Kaper, *Heterogeneous domain decomposition to solve elliptic singular perturbation problems*, SIAM J. Numer. Anal., 34 (1997), to appear.

[14] M. Garbey and J. S. Scroggs, *Asymptotic induced numerical methods for conservation laws*, in Asymptotic Analysis and The Numerical Solution of Partial Differential Equations, H. G. Kaper and M. Garbey, eds., Lecture Notes in Pure and Appl. Math., 130, Marcel Dekker, New York, 1991 pp. 75–96.

[15] N. Glinsky, *Simulation Numérique d'Ecoulement Hypersonique Reactifs Hors Equilibre Chimique*, Thèse, Université de Nice, Nov. 1990.

[16] R. Glowinsky, G. H. Golub, G. A. Meurant, and J. Periaux, in First International Conference on Domain Decomposition Methods for Partial Differential Equations, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1988.

[17] F. A. Howes and G. W. Hedstrom, *A domain decomposition method for a convection-diffusion equation with turning points*, in Domain Decomposition Methods, T. Chan, R. Glowinski, J. Periaux, and O. Widlund, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1989, pp. 38–46.

[18] L. Kang, *The Schwarz Algorithm*, Tech. rep., Wuhan University, Wuhan, China, 1981.

[19] *Asymptotic analysis and the numerical solution of partial differential equations*, H. G. Kaper and M. Garbey eds., Lecture Notes in Pure and Appl. Math., 130, Marcel Dekker, New York, 1991.

[20] *Asymptotic and numerical methods for partial differential equations with critical parameters*, H. G. Kaper and M. Garbey, eds., NATO Adv. Sci. Inst. Ser. C Math. Phys. Sci., 384, Kluwer Academic Publishers, Dordrecht, the Netherlands, 1993.

[21] M. K. Kadalbajoo and Y. N. Reddy, *Asymptotic and numerical analysis of singular perturbation problems: A survey*, Appl. Math. Comput., 30 (1989), pp. 223–259.

[22] Y. Kuznetsov, *Overlapping domain decomposition methods for FE-problems with elliptic singular perturbed operators*, R. Glowinski, Y. A. Kuznetsov, G. Meurant, J. Périaux, and O. B. Widland, eds., in Fourth International Conference on Domain Decomposition for Partial Differential Equations, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1991, pp. 223–241.

[23] P.-L. Lions, *On the Schwarz alternating method*, I, in First International Symposium on Domain Decomposition Methods for Partial Differential Equations, R. Glowinsky, G. Golub, G. Meurant, and J. Periaux, eds, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1988, pp. 1–42.

[24] R. E. O'Malley, *Singular perturbation methods for ordinary differential equations*, Appl. Math. Sci., 89 (1991), Springer Verlag, New York.

[25] B. J. Matkowsky, L. J. Putnick, and G. I. Sivashinsky, *A nonlinear theory of cellular flames*, SIAM J. Appl. Math., 38 (1980), pp. 489–504.

[26] K. Miller, *Numerical analogs to the Schwarz alternating procedure,* Numer. Math., 7 (1965), pp. 91–103.

[27] J. Oliger, W. Skamarock, and W. Tang, *Convergence Analysis and Acceleration of the Schwarz Alternating Method*, Tech. rep., Computer Science Dept., Stanford University, 1986.

[28] J. Pelaez, *Stability of premixed flames with two thin reactions layers*, SIAM J. Appl. Math., 47 (1987), pp. 781–799.

[29] A. Quarteroni, F. Pasquarelli, and A. Vally, *Heterogeneous domain decomposition: Principles, algorithms, applications*, in Fifth International Symposium on Domain Decomposition for Partial Differential Equations, D. E. Keyes et al., eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1991, pp. 129–150.

[30] G. Rodrigue and J. Simon, *A generalization of the numerical Schwarz algorithm*, in Computing Methods in Applied Sciences and Engineering VI, R. Glowinski and J. Lions, eds., North-Holland, Amsterdam, New York, Oxford, 1984, pp. 273–283.

[31] W. P. Tang, *Numerical solution of a turning point problem*, in Fifth International Symposium on Domain Decomposition for Partial Differential Equations, D. E. Keyes et al., eds. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1991, pp. 330–338.

[32] W. P. Tang, *Generalized Schwarz splitting*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 573–595.

SIAM J. SCI. COMPUT.
Vol. 17, No. 5, pp. 1202–1223, September 1996

AN INVESTIGATION OF INTERIOR-POINT ALGORITHMS FOR THE LINEAR TRANSPORTATION PROBLEM*

L. PORTUGAL†, F. BASTOS‡, J. JÚDICE§, J. PAIXÃO¶, AND T. TERLAKY‖

Abstract. Recently, Resende and Veiga [*SIAM J. Optim.*, 3 (1993), pp. 516–537] proposed an efficient implementation of the dual affine (DA) interior-point algorithm for the solution of linear transportation models with integer costs and right-hand-side coefficients. This procedure incorporates a preconditioned conjugate gradient (PCG) method for solving the linear system that is required in each iteration of the DA algorithm. In this paper, we introduce an incomplete QR decomposition (IQRD) preconditioning for the PCG algorithm. Computational experience shows that the IQRD preconditioning is appropriate in this instance and is more efficient than the preconditioning introduced by Resende and Veiga. We also show that the primal dual (PD) and the predictor corrector (PC) interior-point algorithms can also be implemented by using the same type of technique. A comparison among these three algorithms is included and indicates that the PD and PC algorithms are more appropriate for the solution of transportation problems with well-scaled cost and right-hand-side coefficients and assignment problems with poorly scaled cost coefficients. On the other hand, the DA algorithm seems to be more efficient for assignment problems with well-scaled cost coefficients and transportation problems whose cost coefficients are badly scaled.

Key words. linear transportation models, interior-point algorithms, preconditioned conjugate gradient methods, large-scale problems

AMS subject classifications. 90C08, 90C06, 65F10

1. Introduction. Let $O = \{1, \dots, n\}$ and $D = \{1, \dots, m\}$ be the sets of origins and destinations of a product to be shipped. The transportation problem (TP) consists of finding the quantities x_{ij} that should be shipped from each origin $i \in O$ to each destination $j \in D$ in such a way that the total transportation cost is minimized. The TP problem is usually stated as the following linear program:

$$(1)\qquad \begin{aligned} &\min \sum_{i\in O}\sum_{j\in D} c_{ij}x_{ij} \\ &\text{subject to } \sum_{j\in D} x_{ij} = a_i, \qquad i \in O, \\ &\qquad\qquad \sum_{i\in O} x_{ij} = b_j, \qquad j \in D, \\ &\qquad\qquad x_{ij} \ge 0, \quad i \in O, \quad j \in D, \end{aligned}$$

where c_{ij} is the unitary transportation cost from the origin i to the destination j and a_i, b_j represent the quantities available in origin i and required at destination j, respectively. Transportation models have become quite popular during the past several years [32]. Many extensions of the TP problem have been mentioned in the literature. The assignment problem (AP) should be distinguished in this context, as it can be stated in the form (1) with $m = n$ and $a_i = b_j = 1$ for all i and j.

*Received by the editors November 3, 1993; accepted for publication (in revised form) April 7, 1995.

†Departamento de Ciências da Terra, Universidade de Coimbra, 3000 Coimbra, Portugal (luispt@dragao.mat.uc.pt).

‡Departamento de Matemática, Universidade Nova de Lisboa, 2825 Monte Caparica, Portugal (bastos@gist.deio.fc.ul.pt).

§Departamento de Matemática, Universidade de Coimbra, 3000 Coimbra, Portugal (judice@lince.mat.uc.pt).

¶Departamento de Informática, Universidade de Lisboa, 1700 Lisboa, Portugal (paixao@ul.pt).

‖Faculty of Technical Mathematics and Informatics, Delft University of Technology, P.O. Box 5031, 2600 GA Delft, The Netherlands (t.terlaky@dutiwv.twi.tudelft.nl). This author was on leave from Eötvös University, Budapest, Hungary and was partially supported by OTKA 2116.

If we assume without loss of generality that $\sum_{i\in O} a_i = \sum_{j\in D} b_j$, the formulation (1) of the dense TP problem contains exactly one redundant constraint. By dropping this constraint (for instance, the last), we can write the TP problem in the form

$$\begin{aligned} &\min\ c^T x \\ &\text{subject to}\ \ Ax = d, \\ &\qquad\qquad\quad x \geq 0, \end{aligned} \tag{2}$$

where $d \in \mathbb{R}^{m+n-1}$, $x, c \in \mathbb{R}^{mn}$, and $A \in \mathbb{R}^{(m+n-1)\times(mn)}$ has full row rank.

After the introduction of slack variables, the dual of the TP problem (2) can be stated as

$$\begin{aligned} &\max\ d^T y \\ &\text{subject to}\ \ A^T y + s = c, \\ &\qquad\qquad\quad s \geq 0, \end{aligned} \tag{3}$$

where $y \in \mathbb{R}^{n+m-1}$ and $s \in \mathbb{R}^{mn}$.

We note for completeness that a primal feasible solution x is called an *interior point* of the primal problem if $x > 0$, and a dual feasible solution (y, s) is called an *interior point* of the dual problem if $s > 0$.

It is well known that the TP problem can be seen as a minimum-cost network flow problem on a bipartite connected graph [32]. Furthermore, there exists a one-to-one correspondence between basic solutions of the TP problem (2) and spanning trees of this graph. Based on these facts, Resende and Veiga [35, 34] developed an efficient implementation of the dual affine (DA) interior-point algorithm [8] for the solution of large-scale TP problems where the vectors c and d have integer components. This procedure incorporates a preconditioned conjugate gradient (PCG) algorithm [9] to find the ascent direction that is required in each iteration of the DA algorithm.

Resende and Veiga [35, 34] proposed and tested a diagonal and a maximum spanning tree (MST) preconditioning to the PCG algorithm. Extensive computational experience has shown that the diagonal preconditioning is recommended in the very first iterations of the DA algorithm. Then, the MST preconditioning should be used until the end. Because of this observation, they proposed a switch from one preconditioning to the other in the first iterations of the DA algorithm. This type of implementation has worked in practice and seems to be competitive with more traditional techniques that have been designed for the solution of large-scale TP problems [35].

Despite these benefits, the implementation suffers from two drawbacks. First, both preconditionings lose effectiveness during the intermediate iterations of the DA algorithm. Furthermore, the switching point is not known in advance, hence this change can be made earlier or later than desirable.

In this paper we propose an incomplete QR decomposition (IQRD) preconditioning with the objective of overcoming these two drawbacks. Computational experience presented in this paper shows that this technique is, in general, superior to the procedure introduced by Resende and Veiga. Furthermore, we also develop an implementation of the same type for the primal dual (PD) [29, 30, 26] and predictor corrector (PC) [23, 27, 7] interior-point algorithms. A comparative study of these three methods on large-scale TP and AP problems is also presented. This study indicates that the PD and PC algorithms are more appropriate for the solution of TP problems with well-scaled cost and right-hand-side coefficients and assignment problems with poorly scaled cost coefficients. On the other hand, the DA algorithm seems to be more efficient for AP problems with well-scaled cost coefficients and transportation problems whose cost coefficients are badly scaled. The PC algorithm usually takes a smaller number of iterations than the remaining techniques. However, the need of solving two linear systems in each iteration and the use of an iterative method for this purpose renders the PC algorithm, in many cases, less appropriate than the PD method for TP and AP problems.

The paper is organized as follows. Section 2 contains a short description of the DA, PD, and PC interior-point algorithms. Implementation issues are discussed in §3. Finally, the computational experience is described in the last section of the paper.

2. Interior-point algorithms. Since Karmarkar [15] in 1984 presented his polynomial time algorithm for linear programming, hundreds of papers have been published in the field of interior-point methods (see [20] for references). Now, it is, in general, accepted that interior-point methods are strong competitors of the simplex method in solving practical problems and are, in many cases, more efficient for large-scale problems [6, 21, 22, 24]. It is beyond the scope of this paper to discuss and classify the literature of interior-point methods. In this section we restrict ourselves to briefly describing the three interior-point algorithms (the DA scaling, the PD, and the PC) considered in this paper. We include some additional references for the interested reader.

2.1. DA-scaling algorithm. The affine-scaling algorithm [8, 5, 36, 37] was the first interior-point algorithm shown to be competitive [1, 3] with the simplex method for the solution of large-scale linear programs. The dual form of this method was first introduced by Adler et al. [3], and is the more computationally suitable form of the affine-scaling algorithm. The DA method finds an optimal solution by solving the linear program in the standard dual form (3). Having a dual interior starting point, the DA method can be formally presented as follows.

```
DA ALGORITHM.
Given $y^0 \in \{y : A^T y < c\}$, $\delta \in ]0, 1[$,
set $k = 0$.
Repeat
    $s^k = c - A^T y^k.$
    $S_k = \mathrm{diag}(s_1^k, \ldots, s_{mn}^k).$
    $\Delta y = (A S_k^{-2} A^T)^{-1} d.$
    $\Delta s = -A^T \Delta y.$
    $\lambda = \min_{i=1,\ldots,mn}\{-s_i^k/\Delta s_i : \Delta s_i < 0\}.$
    $y^{k+1} = y^k + \delta\lambda\Delta y.$
    $x^{k+1} = -S_k^{-2}\Delta s.$
    $k = k + 1.$
until stopping criterion is verified.
```

The parameter δ is used to guarantee that, in each iteration, y^{k+1} is in the interior of the dual feasible region. The DA algorithm is believed not to be polynomial [2] and its convergence (and the convergence of the primal estimates) depends on the parameter δ. Although it has been shown that δ should be chosen smaller than .66 for good theoretical properties [31, 11], we follow the suggestion of many authors [1, 3, 25, 28] to use $\delta = 0.95$ at every iteration.

2.2. PD algorithm. The so-called PD interior-point methods currently enjoy the most popularity [24]. They were first introduced as path-following methods [18, 29], but they have been extended to potential reduction and logarithmic barrier approaches [19] that are more practical. However, depending on the step size, they still have a bound of $O(\sqrt{n}L)$ and $O(nL)$ iterations (see, also, [12, 30, 26]).

As their name indicates, the PD algorithms operate simultaneously on the primal (2) and the dual (3) problems. Having an initial PD interior-point pair, the search direction in these methods is given as the Newton direction associated with the following asymmetric system of equations that defines the so-called central path:

$$\begin{aligned} Ax &= d, & x &\geq 0, \\ A^T y + s &= c, & s &\geq 0, \\ x_i s_i &= \mu & &\text{for all } i. \end{aligned}$$

The search direction can also be derived as a descent direction for the PD logarithmic barrier function.

As usual, the practical implementation of the PD algorithm slightly differs from the theoretical polynomial variant. The main differences between these two versions rely on the updating of the parameter μ and on the selection of the step sizes (α_p, α_d). If a primal and dual interior starting point is at hand, the practical version of the PD method may be stated in the following form.

```
PD ALGORITHM.
Given $(x^0, y^0, s^0) \in \{(x, y, s) : A^T x = d,\ A^T y + s = c,\ x > 0,\ s > 0\}$, $\delta \in ]0, 1[$,
set $k = 0$.
**Repeat**
    $\mu = (c^T x^k - d^T y^k)/10mn = (x^k)^T s^k/10mn.$
    $X_k = \text{diag}(x_1^k, \dots, x_{mn}^k)$ and $S_k = \text{diag}(s_1^k, \dots, s_{mn}^k).$
    $\Delta y = -(A S_k^{-1} X_k A^T)^{-1} A S_k^{-1}(X_k S_k e - \mu e).$
    $\Delta s = -A^T \Delta y$ and $\Delta x = S_k^{-1}(X_k S_k e - \mu e) - S_k^{-1} X_k \Delta s.$
    $(\alpha_p, \alpha_d) = (\min_{\alpha_i > 0}(x_i - \alpha_i \Delta x_i \geq 0), \min_{\alpha_i > 0}(s_i - \alpha_i \Delta s_i \geq 0)).$
    $x^{k+1} = x^k - \delta\alpha_p \Delta x$, $y^{k+1} = y^k - \delta\alpha_d \Delta y$ and $s^{k+1} = s^k - \delta\alpha_d \Delta s.$
    $k = k + 1.$
**until** stopping criterion is verified.
```

As before, the parameter δ is introduced to assure that $(x^{k+1}, y^{k+1}, s^{k+1}) \in \{(x, y, s) : A^T x = d,\ A^T y + s = c,\ x > 0,\ s > 0\}$. The value $\delta = 0.99995$ has been suggested in the literature [23] and we have followed this recommendation in our experiments. In our implementation we use different step sizes (α_p, α_d) in the primal and dual space, although the use of such parameters is, theoretically, not fully justified yet.

2.3. PC algorithm. The PC algorithm [23, 27] is a technique that has been designed to improve the numerical efficiency of the PD algorithm. In each iteration a descent direction is found by a two-step procedure. In the first step the direction $(\Delta x, \Delta y, \Delta s)$ is computed as in the PD algorithm with $\mu = 0$ (this direction is the so-called PD affine-scaling direction [30]). This direction is used to obtain the value of the so-called barrier parameter μ. The corrector direction $(\Delta x^c, \Delta y^c, \Delta s^c)$ is then computed as a function of this parameter μ. The steps of the PC algorithm are stated below.

```
PC ALGORITHM.
Given $(x^0, y^0, s^0) \in \{(x, y, s) : A^T x = d,\ A^T y + s = c,\ x > 0,\ s > 0\}$, $\delta \in ]0, 1[$,
set $k = 0$.
Repeat
    $X_k = \mathrm{diag}(x_1^k, \ldots, x_{mn}^k)$ and $S_k = \mathrm{diag}(s_1^k, \ldots, s_{mn}^k)$.
    $\Delta y = -(A S_k^{-1} X_k A^T)^{-1} d$.
    $\Delta s = -A^T \Delta y$ and $\Delta x = X_k e - S_k^{-1} X_k \Delta s$.
    $(\alpha_p, \alpha_d) = (\min_{\alpha_i > 0}(x_i - \alpha_i \Delta x_i \geq 0), \min_{\alpha_i > 0}(s_i - \alpha_i \Delta s_i \geq 0))$.
    $\hat{g} = (x - \delta \alpha_p \Delta x)^T (s - \delta \alpha_d \Delta s)$.
    compute $\mu$.
    $\Delta y^c = -(A S_k^{-1} X_k A^T)^{-1} A S_k^{-1} (X_k S_k e - \mu e + \Delta x \Delta s)$.
    $\Delta s^c = -A^T \Delta y^c$ and $\Delta x^c = S_k^{-1}(X_k S_k e - \mu e + \Delta x \Delta s) - S_k^{-1} X_k \Delta s^c$.
    $(\alpha_p, \alpha_d) = (\min_{\alpha_i > 0}(x_i - \alpha_i \Delta x_i{}^c \geq 0), \min_{\alpha_i > 0}(s_i - \alpha_i \Delta s_i{}^c \geq 0))$.
    $x^{k+1} = x^k - \delta \alpha_p \Delta x^c$, $y^{k+1} = y^k - \delta \alpha_d \Delta y^c$ and $s^{k+1} = s^k - \delta \alpha_d \Delta s^c$.
    $k = k + 1$.
until stopping criterion is verified.
```

As in the PD algorithm we use $\delta = 0.99995$ in the implementation of the PC algorithm. According to the recommendations stated in [7], the parameter μ is chosen in the following adaptive way:

$$\mu = \begin{cases} \frac{\hat{g}^3}{mn((x^k)^T s^k)^2} & \text{if } (x^k)^T s^k \geq 1 \\ \frac{(x^k)^T s^k}{\psi(mn)} & \text{if } (x^k)^T (s^k) < 1 \end{cases} \text{, where } \psi(mn) = \begin{cases} (mn)^2 & \text{if } mn \geq 5000 \\ (mn)^{\frac{3}{2}} & \text{if } mn < 5000. \end{cases}$$

The choice of the initial point and the stopping criterion are two important issues in interior-point techniques that have not been discussed yet. The second topic is addressed in the next section. By exploiting the special structure of the TP problem, it is easy to see that if $\alpha = \min\{c_{ij},\ i \in O,\ j \in D\}$, then

$$x_{ij} = \frac{a_i b_j}{\sum_{i \in O} a_i} \quad i \in O,\ j \in D \quad \text{and} \quad y_i = \frac{\alpha}{2} - 1,\ \ i = 1, \ldots, n + m - 1$$

lead to a primal feasible $x > 0$ interior solution and a dual feasible $s > 0$ interior solution. These solutions have been used as starting points in our implementations of the algorithms.

3. Implementation issues. In this section we discuss the implementation of the three algorithms introduced in the previous section for the solution of large-scale transportation problems. It follows from their descriptions that the main computational effort in each iteration relies on the solution of the linear system

$$AGA^T u = t, \tag{4}$$

where $G \in \mathbb{R}^{(mn) \times (mn)}$ and $t \in \mathbb{R}^{n+m-1}$ are defined below.

- $G = S_k^{-2}$ and $t = d$ in the algorithm DA.
- $G = S_k^{-1} X_k$ and $t = A S_k^{-1}(X_k S_k e - \mu e)$ in the algorithm PD.

- $G = S_k^{-1}X_k$ and $t = d$ in the algorithm PC (predictor phase).
- $G = S_k^{-1}X_k$ and $t = AS_k^{-1}(X_kS_ke - \mu e + \Delta x \Delta s)$ in the algorithm PC (corrector phase).

It is easy to see that the explicit computation of AGA^T leads to an almost dense matrix. So the use of a direct method for the solution of the system (4) is not appropriate when m and n are reasonably large. Since AGA^T is a symmetric positive definite matrix, A is sparse and G is diagonal, the PCG algorithm seems to be a good alternative for the solution of the system. In fact, this algorithm only requires sums and scalar products of vectors and matrix-by-vector products. Furthermore, this latter type of operation can be performed without the explicit computation of the matrix AGA^T. The choice of the preconditioning matrix M is the most important issue in the design of such an algorithm. A PCG algorithm for the solution of the system AGA^T is first discussed in this section. Then, we introduce an IQRD preconditioning technique that improves the iterative linear system solver.

3.1. The PCG method. This method attains to find the solution of the linear system $M^{-1}(AGA^T)u = M^{-1}t$, where M is the so-called preconditioning matrix. This matrix should be chosen in such a way that the condition number of $M^{-1}(AGA^T)$ is smaller than that of AGA^T. We discuss later how this matrix can be chosen for the particular instance of the system (4). The steps of the PCG algorithm are presented below.

PCG ALGORITHM.

```
$u^0 = M^{-1}t.$
$r^0 = t - (AGA^T)u^0.$
$z^0 = M^{-1}r^0.$
$p^0 = z^0.$
**Repeat**
    $q^i = (AGA^T)p^i.$
    $\alpha_i = (z^i)^T r^i / (p^i)^T q^i.$
    $u^{i+1} = u^i + \alpha_i p^i.$
    $r^{i+1} = r^i - \alpha_i q^i.$
    $z^{i+1} = M^{-1}r^{i+1}.$
    $\beta_i = (z^{i+1})^T r^{i+1} / (z^i)^T r^i.$
    $p^{i+1} = z^{i+1} + \beta_i p^i.$
    $i = i + 1.$
**until** stopping criterion is verified.
```

It follows from the description of the PCG algorithm that the main issues of this procedure are the stopping criterion, the computation of the product $(AGA^T)\alpha$ for a given vector α, and the choice of the preconditioning matrix M. Next, we discuss these three points.

Stopping criterion. The stopping criterion (*crit* 1) $(z^i)^T r^i < \varepsilon$, where ε is a small tolerance, has been commonly recommended in the PCG algorithm [10]. Its dependence on scaling has suggested more elaborate forms that do not share this drawback. Resende and Veiga [35] proposed the following criterion:

$$\left| 1 - \frac{|t^T(AGA^T)u^i|}{\|t\|_2 \|(AGA^T)u^i\|_2} \right| < \varepsilon. \tag{5}$$

This criterion might be expensive since the quantity $(AGA^T)u^i$ has to be computed. However, it follows from the steps of the PCG algorithm that $AGA^Tu^i = t - r^i$, where r^i is the residual given by the PCG algorithm. So (5) can be replaced by

$$\left| 1 - \frac{|t^T(t-r^i)|}{||t||_2||t-r^i||_2} \right| < \varepsilon. \tag{6}$$

Furthermore, as r^i tends to lose precision as the algorithm progresses, it is advisable to use the criterion (5) after (6) is satisfied.

Hence, we propose the following stopping criterion (*crit* 2):

```
if condition (6) is verified then
    if condition (5) is verified then stop.
    else continue.
else continue.
```

In [4] the following criterion has been proposed:

$$\frac{||r||_\infty}{||AGA^T||_\infty ||u^i||_1 + ||t||_\infty} < \varepsilon, \tag{7}$$

where $r = t-(AGA^T)u^i$. As before, we substitute r by r^i in order to reduce the computational effort of this procedure and use the exact residual when the criterion (7) is satisfied with $r = r^i$.

Hence, we propose the following stopping criterion (*crit* 3):

```
if condition (7) is verified for $r = r^i$ then
    if condition (7) is verified for $r = t - (AGA^T)u^i$ then stop.
    else continue.
else continue.
```

In the last section we discuss the effectiveness of these criteria in the implementation of the interior-point algorithms that have been introduced in the previous section. In particular, special attention will be devoted to the value of the tolerance ε.

The product $\gamma = AGA^T\alpha$. As suggested in [35], the product $\gamma = AGA^T\alpha$ is computed by

$$\begin{aligned} &\text{(i)} \quad \bar{\alpha} = A^T\alpha, \\ &\text{(ii)} \quad \hat{\alpha} = G\bar{\alpha}, \text{ and} \\ &\text{(iii)} \quad \gamma = A\hat{\alpha}. \end{aligned}$$

By doing this, there is no need for computing explicitly the matrix AGA^T, and the sparsities of the matrices A and G are fully exploited in this operation.

Preconditioning. The diagonal preconditioning $M = \text{diag}(AGA^T)$ is simple to construct and has been recommended by many authors [10, 38, 35, 34]. In the context of interior-point algorithms for network flows (as well as for general linear programming), the diagonal preconditioning was first used by Yeh [38]. Computational experience described in [35, 34] has shown that this type of preconditioning may be useful in the first iterations of the DA algorithm but tends to lose effectiveness as the DA algorithm progresses.

This numerical evidence has motivated the search for other forms of preconditionings [35, 34, 13]. It is well known that at least an optimal solution of a linear program is attained at an extreme point of its feasible set, that is, it is a basic feasible solution. For the TP problem each basic solution corresponds to a spanning tree of the graph associated with the model (1). Therefore, we can associate each solution used by the DA algorithm to a partition of the form $A = \begin{bmatrix} B & N \end{bmatrix}$, where B is a basis matrix corresponding to a MST of the graph. In our experiments we define this MST using the diagonal elements of the current matrix G as edge weights.

Let B be the basis matrix associated with such a tree. Then, the MST preconditioning takes the form $M = BG_BB^T$, where G_B is the diagonal matrix of order $n + m - 1$ whose diagonal elements are the edge weights of the tree.

It is important to notice that at the last stages of the DA algorithm the matrix M is quite close to AGA^T, since the values of the diagonal elements of G that are not in G_B are, in general, quite small. Furthermore, B differs from a triangular matrix by a permutation matrix, hence the solution of a system with M simply amounts to solving two triangular systems and performing $n+m-1$ divisions. These two properties explain why the MST preconditioning is quite suitable for this PCG algorithm.

The matrix M may be quite different from AGA^T when all the diagonal entries of G are sufficiently positive. This actually occurs in the first iterations of the DA algorithm. To overcome this fact, Resende and Veiga [35, 34] have advocated a *hybrid scheme* in which diagonal preconditioning is employed in the first iterations of the DA algorithm and MST preconditioning is used from then on. This procedure has been shown to work well in practice but has two drawbacks that are presented below.

(i) The switching point is not known in advance.

(ii) The computational effort of the PCG algorithm is usually large in the intermediate iterations of the DA algorithm, namely, in the last iterations where the diagonal preconditioning is used and in the first iterations in which the MST preconditioning is employed.

In the next subsection we propose a preconditioning technique that overcomes these disadvantages.

3.2. An IQRD technique. Suppose that we are at iteration k of the DA algorithm and let

$$G = \begin{bmatrix} G_B & \\ & G_N \end{bmatrix},$$

where $G_B \in \mathbb{R}^{(n+m-1)\times(n+m-1)}$ is the submatrix of G containing the edge weights of a MST of the graph that represents the TP problem. Furthermore, assume that B is the basis matrix associated with such a tree. If N represents the remaining columns of A, we can write

$$AGA^T = \begin{bmatrix} B & N \end{bmatrix} \begin{bmatrix} G_B & \\ & G_N \end{bmatrix} \begin{bmatrix} B^T \\ N^T \end{bmatrix} = \begin{bmatrix} BG_B^{\frac{1}{2}} & NG_N^{\frac{1}{2}} \end{bmatrix} \begin{bmatrix} G_B^{\frac{1}{2}}B^T \\ G_N^{\frac{1}{2}}N^T \end{bmatrix}.$$

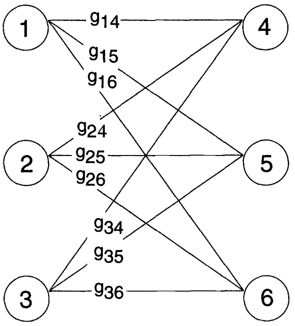
Now, there exist permutation matrices P_c and P_r such that $P_r G_B^{\frac{1}{2}} B^T P_c$ is a triangular matrix. Suppose that $P_c = P_r = I_{(n+m-1)}$ without loss of generality. Then, the Cholesky factorization of the matrix AGA^T can be found by simply computing the QR factorization of the matrix

$$\bar{A} = \begin{bmatrix} G_B^{\frac{1}{2}}B^T \\ G_N^{\frac{1}{2}}N^T \end{bmatrix}. \tag{8}$$

In fact, if $Q\bar{A} = R$, then $AGA^T = \bar{A}^T\bar{A} = R^TQ^TQR = R^TR$.

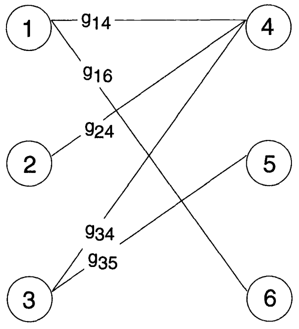
The computation of the QR factorization is not recommended in this instance, since it is not cheaper to compute and destroys the sparsity of the matrix $\bar{A}$. Instead, we propose the computation of an incomplete QR decomposition of the matrix $\bar{A}$ given by (8). In this procedure all the elements of $G_N^{\frac{1}{2}}N^T$ become null by using the diagonal elements of the matrix $G_B^{\frac{1}{2}}B^T$. Furthermore, no fill-in is considered during the entire factorization. It is easy to see that Givens rotations are particularly attractive in this instance. After the factorization, we have $M = FDF^T$, where F is a lower triangular matrix with a diagonal of ones and D is a diagonal matrix with positive diagonal elements. The following example illustrates how this incomplete QR factorization works. Consider the following matrix $G^{\frac{1}{2}}A^T$ and its associated graph obtained from a TP problem after dropping the flow conservation constraint corresponding to node 6:

$$G^{\frac{1}{2}}A^T = \begin{bmatrix}
\sqrt{g_{14}} & & & \sqrt{g_{14}} & \\
\sqrt{g_{15}} & & & & \sqrt{g_{15}} \\
\sqrt{g_{16}} & & & & \\
& \sqrt{g_{24}} & & \sqrt{g_{24}} & \\
& \sqrt{g_{25}} & & & \sqrt{g_{25}} \\
& \sqrt{g_{26}} & & & \\
& & \sqrt{g_{34}} & \sqrt{g_{34}} & \\
& & \sqrt{g_{35}} & & \sqrt{g_{35}} \\
& & \sqrt{g_{36}} & &
\end{bmatrix}$$



Furthermore, consider the following maximum spanning tree and its related matrix $\bar{A}$:

$$\bar{A} = \left[\begin{array}{ccccc|l}
\sqrt{g_{35}} & \sqrt{g_{35}} & & & & \\
& \sqrt{g_{34}} & & \sqrt{g_{34}} & & \\
& & \sqrt{g_{24}} & \sqrt{g_{24}} & & G_B^{\frac{1}{2}}B^T \\
& & & \sqrt{g_{14}} & \sqrt{g_{14}} & \\
& & & & \sqrt{g_{16}} & \\
\cdots & \cdots & \cdots & \cdots & \cdots & \\
\sqrt{g_{15}} & & & & \sqrt{g_{15}} & \\
\sqrt{g_{25}} & & \sqrt{g_{25}} & & & \\
& & \sqrt{g_{26}} & & & G_N^{\frac{1}{2}}N^T \\
& \sqrt{g_{36}} & & & &
\end{array}\right]$$



We start by eliminating the element (6, 1) of the matrix $\bar{A}$ by using the first diagonal element of $G_B^{\frac{1}{2}}B^T$. We note that the element (6, 5) is not transformed and we do not incur fill-in in the places (1, 5) and (6, 2) in the following matrix 1:

$$\begin{bmatrix}
\sqrt{g_{35}+g_{15}} & \frac{g_{35}}{\sqrt{g_{35}+g_{15}}} & & & \\
& \sqrt{g_{34}} & & \sqrt{g_{34}} & \\
& & \sqrt{g_{24}} & \sqrt{g_{24}} & \\
& & & \sqrt{g_{14}} & \sqrt{g_{14}} \\
& & & & \sqrt{g_{16}} \\
\cdots & \cdots & \cdots & \cdots & \cdots \\
0 & & & & \sqrt{g_{15}} \\
\sqrt{g_{25}} & & \sqrt{g_{25}} & & \\
& & \sqrt{g_{26}} & & \\
& \sqrt{g_{36}} & & &
\end{bmatrix}.$$

Now, by using the last diagonal element of the upper block matrix we turn to zero the element (6, 5) in the following matrix 2:

$$\begin{bmatrix}
\sqrt{g_{35}+g_{15}} & \frac{g_{35}}{\sqrt{g_{35}+g_{15}}} & & & \\
 & \sqrt{g_{34}} & & \sqrt{g_{34}} & \\
 & & \sqrt{g_{24}} & \sqrt{g_{24}} & \\
 & & & \sqrt{g_{14}} & \sqrt{g_{14}} \\
 & & & & \sqrt{g_{16}+g_{15}} \\
\cdots & \cdots & \cdots & \cdots & \cdots \\
0 & & & & 0 \\
\sqrt{g_{25}} & & \sqrt{g_{25}} & & \\
 & & \sqrt{g_{26}} & & \\
 & \sqrt{g_{36}} & & &
\end{bmatrix}.$$

The procedure can be repeated to annihilate the second row of the lower block matrix in the following matrix 3:

$$\begin{bmatrix}
\sqrt{g_{35}+g_{15}+g_{25}} & \frac{g_{35}}{\sqrt{g_{35}+g_{15}+g_{25}}} & & & \\
 & \sqrt{g_{34}} & & \sqrt{g_{34}} & \\
 & & \sqrt{g_{24}+g_{25}} & \frac{g_{24}}{\sqrt{g_{24}+g_{25}}} & \\
 & & & \sqrt{g_{14}} & \sqrt{g_{14}} \\
 & & & & \sqrt{g_{16}+g_{15}} \\
\cdots & \cdots & \cdots & \cdots & \cdots \\
0 & & & & 0 \\
0 & & 0 & & \\
 & & \sqrt{g_{26}} & & \\
 & \sqrt{g_{36}} & & &
\end{bmatrix}.$$

After all the rows of $G_N^{\frac{1}{2}}N^T$ are eliminated we have

$$P_r D^{\frac{1}{2}} F^T P_c$$

$$= \begin{bmatrix}
\sqrt{g_{35}+g_{15}+g_{25}} & \frac{g_{35}}{\sqrt{g_{35}+g_{15}+g_{25}}} & & & \\
 & \sqrt{g_{34}+g_{36}} & & \frac{g_{34}}{\sqrt{g_{34}+g_{36}}} & \\
 & & \sqrt{g_{24}+g_{25}+g_{26}} & \frac{g_{24}}{\sqrt{g_{24}+g_{25}+g_{26}}} & \\
 & & & \sqrt{g_{14}} & \sqrt{g_{14}} \\
 & & & & \sqrt{g_{16}+g_{15}}
\end{bmatrix}.$$

Now, if we repermute the rows and columns of this last matrix to the original ordering, we obtain

$$D^{\frac{1}{2}}F^T = \begin{bmatrix} \sqrt{g_{16}+g_{15}} & & & & \\ & \sqrt{g_{24}+g_{25}+g_{26}} & & \frac{g_{24}}{\sqrt{g_{24}+g_{25}+g_{26}}} & \\ & & \sqrt{g_{34}+g_{36}} & \frac{g_{34}}{\sqrt{g_{34}+g_{36}}} & \\ \sqrt{g_{14}} & & & \sqrt{g_{14}} & \\ & & \frac{g_{35}}{\sqrt{g_{35}+g_{15}+g_{25}}} & & \sqrt{g_{35}+g_{15}+g_{25}} \end{bmatrix}.$$

For this example,

$$D = \begin{bmatrix} g_{16}+g_{15} & & & & \\ & g_{24}+g_{25}+g_{26} & & & \\ & & g_{34}+g_{36} & & \\ & & & g_{14} & \\ & & & & g_{35}+g_{15}+g_{25} \end{bmatrix}$$

and

$$F^T = \begin{bmatrix} 1 & & & & \\ & 1 & & \frac{g_{24}}{g_{24}+g_{25}+g_{26}} & \\ & & 1 & \frac{g_{34}}{g_{34}+g_{36}} & \\ \frac{g_{14}}{g_{14}} & & & 1 & \\ & & \frac{g_{35}}{g_{35}+g_{15}+g_{25}} & & 1 \end{bmatrix}.$$

We notice that matrices D and F are actually obtained without computing $D^{\frac{1}{2}}F^T$ in order to avoid square root operations. Suppose that the MST is rooted at node r, which is the node corresponding to the flow conservation equation that has been removed from the formulation. Furthermore, let $\mathcal{A}$ denote the subset of arcs belonging to the tree and let $pred(i)$ represent the predecessor node of node i in the tree. Then, the procedure used to compute the nonzero elements of matrix D and the off-diagonal elements of matrix F can be presented as follows:

```
for all nodes $i : i \neq r$ do
    $j = pred(i)$.
    if arc $(i, j) \in \mathcal{A}$ then $d_{ii} = g_{ij}$ else $d_{ii} = g_{ji}$.
for all arcs $(i, j) : (i, j) \notin \mathcal{A}$ do
    if node $i \neq r$ then $d_{ii} = d_{ii} + g_{ij}$.
    if node $j \neq r$ then $d_{jj} = d_{jj} + g_{ij}$.
for all nodes $i : i \neq r$ do
    $j = pred(i)$.
    if node $j \neq r$ then
        if arc $(i, j) \in \mathcal{A}$ then $f_{ij} = g_{ij}/d_{ii}$ else $f_{ji} = g_{ji}/d_{ii}$.
```

It is important to notice that the PCG algorithm discussed in this section can be incorporated in all three interior-point algorithms introduced in the previous section.

In the first iterations of the interior-point algorithms, $D \approx \text{diag}(AGA^T)$ and $F \approx I_{m+n-1}$. In the last iterations of the interior-point algorithms, $G_N \approx 0$ and, consequently, $D \approx G_B$ and $F \approx B$. Thus, we can conclude that the IQRD preconditioning is quite similar to the diagonal and MST preconditionings in the initial and terminal phases of the interior-point methods, respectively. Since the IQRD preconditioning also takes into account the elements of $G_N^{\frac{1}{2}}N$, then it should require fewer conjugate gradient iterations in the intermediate phase of the interior-point algorithms.

The solution of the system with FDF^T requires $O(n+m)$ divisions, $O(n+m)$ multiplications, and $O(m+n)$ subtractions, since D is a diagonal matrix and F can be permuted into a triangular matrix with diagonal elements equal to one. Furthermore, solving a system with the matrix BG_BB^T requires $O(n+m)$ divisions and $O(n+m)$ subtractions. So there is an increase of $O(n+m)$ multiplications in each iteration of the PCG algorithm if the IQRD preconditioning is used instead of the MST procedure. Furthermore, the IQRD preconditioning amounts to an increase of $O(n+m)$ multiplications and $O(n+m)$ subtractions over the diagonal preconditioning. These gaps are not too meaningful, since each computation of a product $AGA^T\alpha$ that is required in each iteration of the PCG algorithm involves $O(nm)$ multiplications and $O(nm)$ sums. It is also important to add that the construction of the preconditioning matrices F and D is done once in each iteration of the interior-point methods and requires $O(nm)$ sums and $O(n+m)$ divisions. This is smaller than the computation of a product $AGA^T\alpha$. All these considerations lead to the conclusion that the IQRD preconditioning is recommended provided it reduces the number of PCG iterations by even a small fraction. This is exactly what occurs in practice in instances from the class considered in our experiments and is well displayed in the computational results presented in the last section of this paper.

3.3. Stopping criteria for the interior-point algorithms. Given a basic solution for the TP problem with a basis matrix B, it is well known that this solution is optimal if it satisfies the following conditions:

$$x_B = B^{-1}d \geq 0, \tag{9}$$

$$c_N - N^T(B^T)^{-1}c_B \geq 0, \tag{10}$$

where N is the matrix corresponding to the columns of A associated with the nonbasic variables. We note that the first criterion is quite cheap to verify, but the second inequality may lead into a large computational effort. As suggested in [35], a further criterion can be useful when the data of the transportation problem is integer. This criterion follows from the duality theory of linear programming and simply states that if x_B satisfies the condition (9), and a dual feasible solution y^k given by any of the interior-point methods satisfies

$$c_B^T x_B - d^T y^k < 1, \tag{11}$$

then $x = (x_B, 0)$ is an optimal solution of the TP problem. This criterion is obviously quite cheap to compute, since y^k is available from the interior-point methods. However, in some instances the criterion (10) is verified without the satisfaction of the inequality (11). These observations lead to the following stopping criterion that is used in the three interior-point methods discussed in the previous section:

Let B be the basis matrix associated with the current MST and y^k be the approximation to y at the iteration k of the interior-point algorithm.

```
**if** $x_B = B^{-1}d \geq 0$ **then**
    **if** $c_B^T x_B - d^T y^k < 1$ **then** stop.
    **else**
        **if** $c_N - N^T(B^T)^{-1}c_B \geq 0$ **then** stop.
        **else** continue.
**else** continue.
```

It is important to notice that this stopping criterion fails to hold if the condition $B^{-1}d \geq 0$ is never satisfied. This situation has never occurred in our computational experiments. However, when the problem is degenerate and the perturbation schemes fail, it is possible that the algorithms may never achieve a MST corresponding to a primal optimal feasible solution. We refer to [33, 34] for a procedure to overcome this difficulty.

4. Computational experience. In this section we first compare the efficiency of the IQRD preconditioning with the hybrid preconditioning (diagonal and MST) introduced by Resende and Veiga [35, 34]. These two techniques are incorporated in the PCG algorithm that is employed in the solution of the linear systems required by the three interior-point methods. In our second experiment we investigate the three stopping criteria for this method discussed in the previous section. We come to some conclusions about the value of the tolerance ε used in these criteria. Our third experiment is devoted to investigating the effect of the scaling in the cost coefficients and right-hand-side elements on the performance of the interior-point algorithms. Finally, we report a computational comparison between the interior-point methods and the network simplex code NETFLO of Kennington and Helgason [16] on problems generated by the standard problem generator NETGEN [17]. In these last two experiments we use the stopping criterion and the tolerance value that are indicated by our second experiment as the most recommended to be incorporated in the interior-point methods.

All the experiments were performed on a SUN SPARCstation 10-52 with 64 Mbytes of RAM. The codes are written in FORTRAN and were compiled with the Sun Fortran Compiler (f77) using flags **-O4 -cg89 -libmil -native**. All the CPU times reported are excluded from input and output time and were measured with the internal function **etime**.

All the test problems described in this paper are dense with a number of arcs equal to mn. In the first three experiments, the test problems are generated by a technique that first considers a spanning tree related with a basic optimal solution. Then, the components of the dual (primal) solution associated with the edges that belong (do not belong) to the tree are fixed to zero. The remaining primal and dual variables are randomly generated in such a way that the elements of the right-hand-side and cost vectors are integers belonging to the intervals $[1, d_{max}]$ (1 for the assignment problem) and $[1, c_{max}]$, respectively. If we wish the solution to be degenerate (as in the assignment problem), then some of these latter variables are also set equal to zero.

As stated before, we considered test problems with nondegenerate and degenerate optimal solutions. Our experience has shown that the stopping criterion loses effectiveness in the second case. In fact, if the dual optimal solution is degenerate, the current maximum spanning tree may be related with an infeasible primal basic solution when the interior-point algorithms converge to the optimal primal face. Furthermore, if the optimal primal solution is degenerate it is also possible to be correctly identified by the current MST without the verification of the stopping criterion. As suggested in [35], we can use an ε-perturbation technique to help to overcome such difficulties. In this procedure we solve a perturbed TP problem where the cost coefficient $\bar{c}$ and right-hand-side $\bar{d}$ vectors are defined by $\bar{c}_j = c_j + \delta_{c_i}\varepsilon$ and $\bar{d}_j = d_j + \delta_{d_j}\varepsilon$, where δ_{c_i} and δ_{d_j} are uniform random values in the interval $[-1, 1]$, and ε is a small perturbation. This technique is far from being perfect despite being, as far as we know, the most effective device in preventing degeneracy effects in practice. If ε is too small, then usually the perturbed problem is still degenerate. If ε is too big, then two cases may be possible. If we use the perturbed data in the termination criterion, the interior-point algorithms may reach a suboptimal solution. On the other hand, if we use the termination criterion with the original vectors c and d, the interior-point algorithms may never stop. In order to avoid obtaining suboptimal solutions, we have set $\varepsilon = \min(10^{-3}, 10^{-6}(m + n - 1))$ and we have used the original data in the stopping criterion, as suggested in Kaliski and Ye [14]. The implementations of this

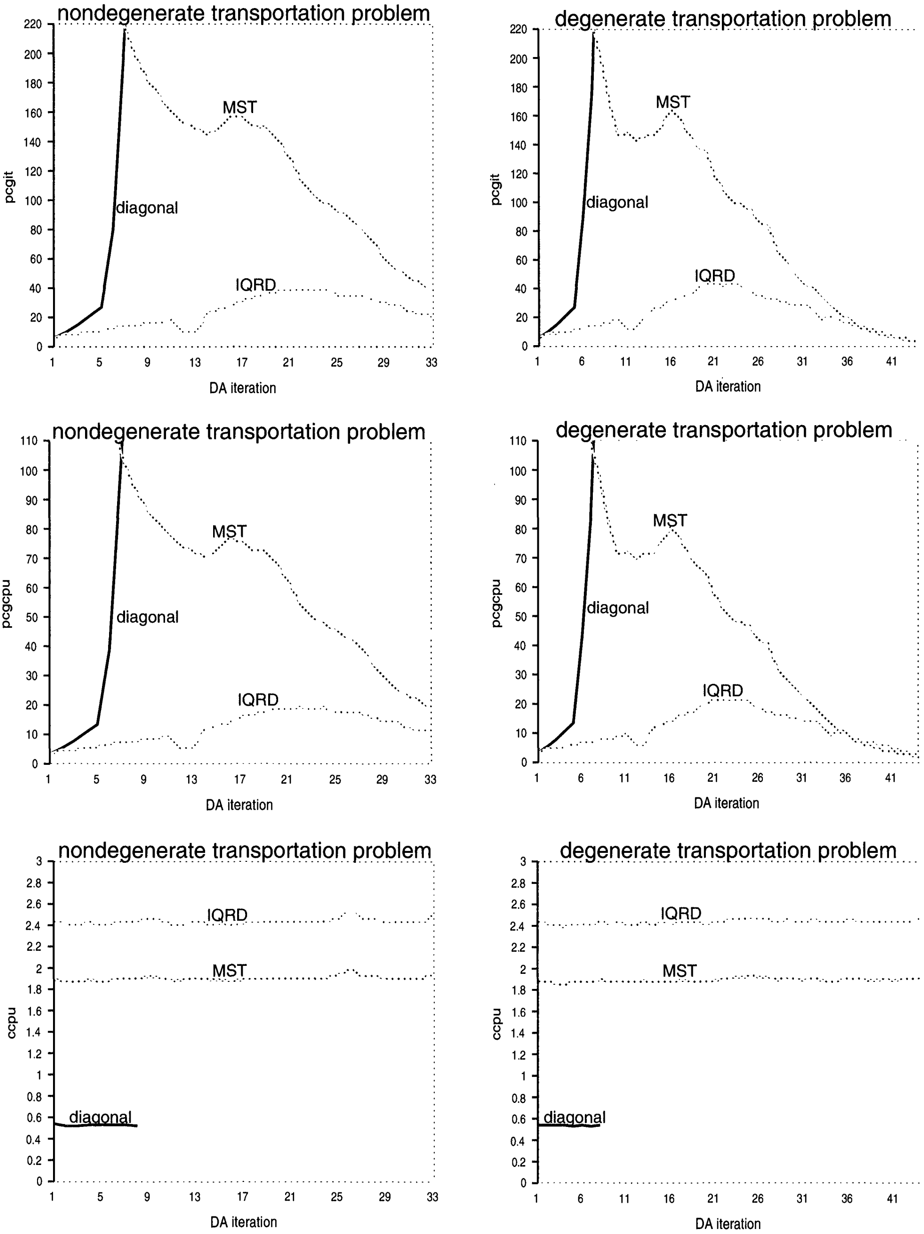


FIG. 1. *Performance of the preconditionings for transportation problems with* $n = m = 1000$.

procedure in our codes have provided an optimal solution for all the test problems described in this paper.

As stated before, we study in our first experiment the efficiency of the IQRD, diagonal, and MST preconditionings for the PCG algorithm that is implemented in the DA method. We have used stopping criterion crit 1 with $\varepsilon = 10^{-6}$. The results of this experiment are illustrated in Figures 1 and 2 and Table 1 for the solution of assignment and general transportation problems with $m = n = 1000$, $c_{max} = 10$, and $d_{max} = 10$. Figures 1 and 2 report each iteration of the DA algorithm and Table 1 reports a comparison between the IQRD preconditioning and the hybrid scheme of Resende and Veiga. In this last experiment we suppose that the optimal

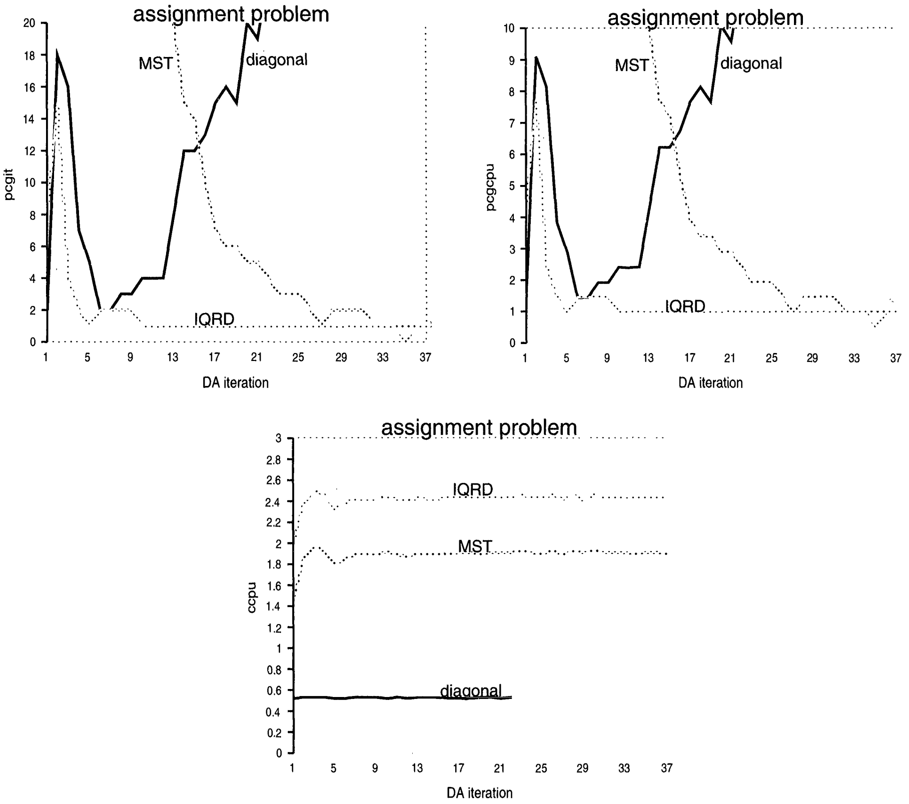


FIG. 2. *Performance of the preconditionings for an assignment problem with $n = m = 1000$.*

TABLE 1

Comparison among the hybrid scheme and the IQRD preconditioning for TP and AP problems with $n = m = 1000$.

	Hybrid scheme				IQRD preconditioning		
Problem	ccpu	pcgit	pcgcpu	optiter	ccpu	pcgit	pcgcpu
Nondegenerate TP	53.39	3447	1676.5	8	80.35	794	400.4
Degenerate TP	73.92	3443	1687.1	8	107.15	982	496.9
AP	49.93	171	100.0	16	89.51	65	49.4

switching iteration for each problem is known in advance, that is, that we know previously the best iteration in which the hybrid scheme should switch from one preconditioning to the other. The symbols **ccpu**, **pcgit**, **pcgcpu**, and **optiter** represent the CPU time in seconds to construct the preconditioners plus the time to compute the MST (this last item is not included if the diagonal preconditioning is used), the number of PCG iterations, the CPU time in seconds required by the PCG method, and the optimal switching iteration for each problem, respectively.

We make the following conclusions:

- The diagonal preconditioning is effective during the initial iterations of the DA algorithm, but its efficiency is lost after this stage. On the other hand the opposite situation occurs with the MST preconditioning. Actually, this confirms the claims presented in [35] that have led to the hybrid scheme proposed by Resende and Veiga.

TABLE 2

Number of PCG iterations required by the interior-point algorithms to solve an AP problem for different values of the tolerance in the stopping criteria for the PCG algorithm ($n = m = 1000$).

	Crit 1			Crit 2			Crit 3		
Tolerance	DA	PD	PC	DA	PD	PC	DA	PD	PC
1.0e−1	1284								
1.0e−3	1589		365						
1.0e−6	2046	781	525						
1.0e−8	2323	1061	810	1105	870	608	1102		
1.0e−10	2606	1323	1041	1461	1092	935	1754		
1.0e−12	2877	1491	1261	1766	1312	1170	2313	888	829
1.0e−14	3139	1653	1267	2063	1474	1183	2851	1403	1250
1.0e−16	3406	1816	1511	2334			3370	1745	1606

TABLE 3

Number of PCG iterations required by the interior-point algorithms to solve a TP problem for different values of the tolerance in the stopping criteria for the PCG algorithm ($n = m = 1000$).

	Crit 1			Crit 2			Crit 3		
Tolerance	DA	PD	PC	DA	PD	PC	DA	PD	PC
1.0e−1	1147	574	865						
1.0e−3	2061	684	981						
1.0e−6	2815	825	1148	587		568			
1.0e−8	3214	921	1270	1253		588			
1.0e−10	3581	1009	1386	1991	482	1024		554	706
1.0e−12	3926	1098	1487	2613	617	997		788	1036
1.0e−14	4310	1180	1600	3030	760	1121	3420	968	1276
1.0e−16	4703	1276	1704		1290		3816	1148	1479

- The IQRD preconditioning usually requires a smaller number of PCG iterations than the diagonal and the MST preconditionings at any stage of the DA algorithm. The gap is bigger in the intermediate iterations of that algorithm.
- The IQRD preconditioning requires a bit more CPU time to construct than the other two preconditionings. However, the IQRD preconditioning is able to reduce sufficiently the number of PCG iterations that compensate for such an additional computational effort. In fact, except in the very first and in the very last DA iterations, the CPU time ccpu+pcgcpu is smaller if the IQRD preconditioning is used than if the diagonal or MST preconditionings are employed.
- For dense TP and AP problems the IQRD preconditioning is preferable to the hybrid scheme of Resende and Veiga to be incorporated in the interior-point methods, even if we suppose that it is possible to know previously the optimal switching iteration. In fact, if we use the IQRD preconditioning instead of the hybrid scheme, we reduce by 77.0%, 71.5%, and 62.0% the number of PCG iterations required to solve the nondegenerate TP problem, the degenerate TP problem, and the AP problem, respectively. The corresponding decrease in the CPU time ccpu+pcgcpu when those test problems are solved is 72.8%, 65.8%, and 7.35%.

Based on the results of the first experiment, we decided to use the IQRD preconditioning in the implementation of the DA, PD, and PC interior-point algorithms. In our second experiment we study the performance of these methods under stopping criteria crit 1, crit 2, and crit 3 discussed in the previous section and a set of different values for the tolerance ε. The results of this experiment are illustrated in Tables 2 and 3, which report the performance of interior-point methods DA, PD, and PC on TP and AP problems with $m = n = 1000$. It is important to add that we chose the test problems for which the interior-point algorithms have the most difficulty in finding their optimal solutions. The data in Tables 2 and 3 represent the number of iterations

TABLE 4
Solution of transportation problems with $c_{max} = 1.8e1$ and $d_{max} = 1.1e1$.

	DA			PD			PC		
m and n	iter	pcgit	cpu	iter	pcgit	cpu	iter	pcgit	cpu
200	13	190	5.69	4	46	1.97	4	102	3.49
400	18	306	37.39	5	63	10.42	5	128	18.03
600	23	472	121.70	8	154	47.54	7	225	66.58
800	30	623	294.50	6	91	57.81	7	199	115.20
1000	34	813	584.20	9	214	175.70	9	325	265.90
total	118	2404	1043.48	32	568	293.44	32	979	469.20

TABLE 5
Solution of transportation problems with $c_{max} = 1.8e1$ and $d_{max} = 2.5e3$.

	DA			PD			PC		
m and n	iter	pcgit	cpu	iter	pcgit	cpu	iter	pcgit	cpu
200	30	550	15.50	25	415	14.32	19	626	18.80
400	42	895	96.65	32	629	82.82	24	914	108.80
600	51	1472	340.60	40	1318	327.20	24	1551	353.60
800	62	1642	713.60	50	1347	658.10	28	1427	645.70
1000	70	2158	1430.00	53	1734	1254.00	30	1803	1230.00
total	255	6717	2596.65	200	5443	2336.44	125	6321	2356.90

of the PCG algorithm for different values of the tolerance ε when the three stopping criteria are incorporated in this procedure. If for a certain criterion there is no number for a value of $\varepsilon = 10^{-p}$, then either this tolerance value is too big for the interior-point method to reach an appropriate direction and, consequently, the method cannot converge to the optimal solution, or it is too small for the PCG algorithm to converge in some iteration of this latter technique.

The results shown in Tables 2 and 3 lead to the following conclusions:

- crit 1 allows a large choice for the value of tolerance ε. As expected, the total number of PCG iterations increases with a decrease of the tolerance value. On the other hand, the incorporation of the other two criteria requires much care in the choice of the value of ε. For instance, the DA algorithm can only work with a value of the tolerance smaller than or equal to 10^{-14} if crit 3 is used. Hence, we can conclude that crit 1 is the most robust among the three criteria discussed in the previous section. This does not mean that crit 1 takes a smaller number of iterations than the other criteria. On the contrary, crit 2 usually leads to the smallest iteration count; crit 3 is also better than crit 1 in this respect.
- When a TP problem is solved, then a value of ε in the interval $[10^{-6}, 10^{-3}]$ is quite safe if crit 1 is used in the implementation of the three interior-point algorithms. The same kind of conclusions can be stated for the solution of AP problems by the DA algorithm. However, a slightly smaller value for ε should be considered ($\varepsilon \in [10^{-8}, 10^{-6}]$) when AP problems are solved by any of the remaining interior-point PD and PC algorithms.

Based on these results, we have decided to use in our third experiment criterion crit 1 for all the interior-point methods. In the DA algorithm we set the value of the tolerance to be equal to 10^{-6}. The same value of ε is employed in the PD and PC algorithms when TP problems are solved. Furthermore, $\varepsilon = 10^{-8}$ is used in these last procedures for the solution of AP problems. As stated before, in the next study we investigate the importance of scaling of the vectors c and d. We considered assignment and general nondegenerate and degenerate transportation problems with integer data, $m = n = 200, 400, 600, 800, 1000$ and different scalings of the vectors d and c. The results of this study are illustrated in Tables 4–9. The symbols **iter**, **pcgit**, and **cpu** represent the number of interior-point iterations, the number of

TABLE 6
Solution of transportation problems with $c_{max} = 1.0e6$ and $d_{max} = 1.1e1$.

	DA			PD			PC		
m and *n*	iter	pcgit	cpu	iter	pcgit	cpu	iter	pcgit	cpu
200	5	106	2.82	10	192	6.46	14	531	14.88
400	6	181	16.86	11	260	31.36	13	609	66.34
600	6	246	51.30	11	360	89.94	15	782	193.80
800	7	328	125.60	16	542	245.40	16	849	380.60
1000	8	317	198.80	15	623	425.10	14	927	617.70
total	32	1178	395.38	63	1977	798.26	72	3698	1273.32

TABLE 7
Solution of transportation problems with $c_{max} = 1.0e6$ and $d_{max} = 2.5e3$.

	DA			PD			PC		
m and *n*	iter	pcgit	cpu	iter	pcgit	cpu	iter	pcgit	cpu
200	9	215	5.61	32	588	19.20	22	886	25.06
400	12	359	35.77	37	797	100.40	31	1461	158.20
600	14	517	110.30	43	1515	365.50	37	2873	623.40
800	15	696	263.90	48	1416	670.50	30	1755	765.20
1000	15	595	380.40	49	1723	1370.00	27	1902	1243.00
total	65	2382	795.98	209	6039	2525.60	147	8877	2814.86

TABLE 8
Solution of assignment problems with $c_{max} = 1.0e1$.

	DA			PD			PC		
m and *n*	iter	pcgit	cpu	iter	pcgit	cpu	iter	pcgit	cpu
200	27	70	6.28	16	230	8.37	12	341	10.79
400	21	58	22.07	21	376	51.33	12	365	46.66
600	26	59	60.68	21	381	120.80	11	389	110.10
800	24	53	100.30	22	369	223.20	13	389	215.60
1000	39	84	259.90	27	519	264.80	16	589	462.30
total	137	324	449.23	107	1875	668.50	64	2073	845.45

TABLE 9
Solution of assignment problems with $c_{max} = 1.0e6$.

	DA			PD			PC		
m and *n*	iter	pcgit	cpu	iter	pcgit	cpu	iter	pcgit	cpu
200	28	438	12.58	20	419	12.65	19	717	20.24
400	34	662	75.74	25	620	72.54	23	967	108.60
600	45	1042	257.00	26	980	199.30	27	1468	353.90
800	41	936	427.80	27	821	383.80	35	1744	795.50
1000	40	1041	723.40	28	961	685.50	28	1576	1101.00
total	188	4119	1496.52	126	3801	1353.79	132	6472	2379.24

PCG iterations, and the CPU time in seconds, respectively. We only report the results with nondegenerate TP problems, since we have observed that the relative performance of the three interior-point algorithms is similar for degenerate and nondegenerate transportation problems.

We have achieved the following conclusions about the scaling of the vectors c and d:

- All three interior-point algorithms seem to be sensitive to bad scaling in the right-hand-side vector d of the transportation problem. However, the PD techniques are usually more effected than the DA algorithm by an increase in d_{max}. In fact, when d_{max} changes from $1.1e1$ to $2.5e3$, the CPU time increases, on average, 327.0% and

196.8% for both PD and PC algorithms, respectively, and only 153.4% for the DA algorithm.

- The PD and PC algorithms seem to be less effected by bad scaling in c than by bad scaling in d. In fact, the computational effort of the PD method grows, on average, 26.4% for TP problems and 102.5% for AP problems when c_{max} changes from $1.8e1$ to $1.0e6$. Furthermore, the average CPU time increasing for the PC algorithm is 44.7% for the transportation problem and 181.4% for the assignment problem. It is interesting to note that the variation of the performance of the DA algorithm with the scaling in c depends on the class of problem. In the transportation problem we obtain an average decrease of 67.27% in CPU time when the value of c_{max} changes from $1.8e1$ to $1.0e6$. On the other hand, for the same change in the scaling of c, that value increases 233.13% when we solve the assignment problem.

In this last experiment we compare the three interior-point methods with the network simplex code NETFLO of Kennington and Helgason [16] on problems generated by the standard problem generator NETGEN [17]. We have considered assignment and transportation problems with integer data, $m = n =$ 200, 400, 600, 800, 1000, number of arcs equal to mn, cost for the arcs in the interval $[1, c_{max}]$ with $c_{max} = 1.0e1, 1.0e6$, and total supply equal to $1.0e2n$ and $1.0e6n$ (equal to n in the assignment problems). The results of this study are illustrated in Tables 10–13. The symbols iter and cpu represent the number of iterations and the CPU time in seconds required by the different algorithms to achieve an optimal solution for the test problems, respectively.

The results displayed in Tables 10–13 show that the network simplex code NETFLO is faster than the interior-point methods in all the test problems. However, one should not draw any final conclusions from these results. First, we have used in the interior-point methods a tighten robust stopping criterion that is not the best for all the problems. Second, we have not worried about the initial point, since the main purpose of the paper was to investigate the efficiency of a new preconditioning and to compare the PD interior-point techniques with the DA method used by Resende and Veiga [35]. Most importantly, instances are dense and too small. As in Resende and Veiga [35], competitiveness is achieved in sparse problems only in instances that are much larger than those considered in this paper. We believe that a more appropriate initial point and a different stopping criterion in which the values of the tolerance in the PCG method change during the progress of the interior-point methods may turn these methods more competitive with the classical algorithms for transportation and assignment problems.

Despite it being difficult to draw final claims about the most efficient interior-point method, the results indicate that the PD and the PC algorithms seem to be the most appropriate techniques for the solution of TP problems when the cost coefficients and right-hand-side vectors c and d are not badly scaled. If c and d are both badly scaled, then the DA algorithm usually performs better than the PD methods. The efficiencies of the three algorithms seem to be similar for the TP problems in which only the vector d is badly scaled. Finally, if only vector c is badly scaled, then, generally, the DA algorithm outperforms the algorithms PD and PC. In the solution of AP problems, bad scaling in the vector c seems to have an opposite effect on the performance of the interior-point methods. In fact, the DA algorithm seems to be more efficient than the PD algorithm when c is well scaled, while the opposite situation occurs in the presence of badly scaled AP problems. The PC method performs worse than the DA and PD algorithms in the solution of AP problems.

Currently the use of the PC algorithm for the solution of large-scale linear programs is well accepted [7]. In fact, this technique usually reduces the overall amount of iterations of the PD algorithm. Furthermore, as a direct solver is used to process the Newton equations, then there is an increase of only two triangular systems per iteration of the PD method. The

TABLE 10

Solution of NETGEN transportation problems with $c_{max} = 1.0e1$ and total supply equal to $1.0e2n$..

	DA		PD		PC		NETFLO	
m and *n*	iter	cpu	iter	cpu	iter	cpu	iter	cpu
200	31	11.66	28	17.14	14	15.34	908	0.78
400	32	62.38	26	68.83	15	65.16	2016	3.02
600	46	206.30	32	200.90	26	286.10	2848	8.00
800	88	560.00	37	498.40	18	379.30	3918	18.17
1000	86	890.16	37	770.00	18	590.40	4671	26.94
total	283	1705.50	160	1555.27	91	1336.30	14451	56.91

TABLE 11

Solution of NETGEN transportation problems with $c_{max} = 1.0e1$ and total supply equal to $1.0e6n$.

	DA		PD		PC		NETFLO	
m and *n*	iter	cpu	iter	cpu	iter	cpu	iter	cpu
200	22	14.51	22	14.29	14	17.09	959	0.65
400	39	107.20	27	86.65	14	76.48	2053	2.82
600	42	300.60	35	259.30	25	299.40	2590	6.45
800	46	656.20	38	512.50	24	573.80	4272	12.41
1000	51	1243.00	45	1071.00	20	847.50	4735	20.95
total	200	2321.51	167	1943.74	97	1814.27	14609	43.28

TABLE 12

Solution of NETGEN assignment problems with $c_{max} = 1.0e1$.

	DA		PD		PC		NETFLO	
m and *n*	iter	cpu	iter	cpu	iter	cpu	iter	cpu
200	22	6.25	16	9.19	11	9.91	281	0.56
400	26	30.49	19	48.35	10	44.85	485	2.66
600	14	42.76	23	163.10	11	108.10	732	6.91
800	15	71.30	25	272.60	11	202.90	1002	10.97
1000	19	152.30	23	387.40	13	343.70	1142	20.48
total	96	303.10	106	876.64	56	709.46	3642	41.58

TABLE 13

Solution of NETGEN assignment problems with $c_{max} = 1.0e6$.

	DA		PD		PC		NETFLO	
m and *n*	iter	cpu	iter	cpu	iter	cpu	iter	cpu
200	24	13.37	21	13.43	19	20.90	2350	0.75
400	30	74.10	24	66.37	24	113.80	9599	5.68
600	34	256.50	26	232.00	24	320.60	16409	18.28
800	38	446.30	29	403.60	24	586.30	30480	37.62
1000	41	821.30	27	644.60	32	1267.00	39178	57.36
total	167	1611.57	127	1360.00	123	2308.60	98016	119.69

results shown in Tables 4–13 seem to confirm the ability of the PC strategy to reduce the total amount of the interior-point iterations. However, the use of an iterative solver almost doubles the computational effort of each iteration of the PD algorithm. This explains why the CPU time is in many cases smaller for the simple PD algorithm. So the PD algorithm may be more appropriate than the PC method for the solution of TP and AP problems when these procedures are implemented in the way described in this paper.

Our final remark is concerned with the competitiveness of the interior-point methods with the classical techniques (which NETFLO represents in this paper) in solving linear minimum-cost network flow problems. Resende and Veiga [35] have already shown that if the parallelism of the PCG algorithm is exploited, the DA method is expected to outperform NETFLO and the relaxation code RELAX in the solution of sufficiently large assignment problems. Later, the same authors also demonstrated that the DA algorithm is more efficient than the classical methods in several classes of linear network flow problems, even when implemented in sequential architectures [34]. In this paper, despite having introduced a new preconditioning that greatly reduces the computational effort required by the PCG algorithm, the results show that NETFLO is still more efficient than the interior-point methods in the solution of dense assignment and transportation problems in sequential architectures. As we stated before, there is much work to be done for the interior-point algorithms to be competitive with the classical techniques. We have already mentioned some possible modifications in the algorithms that may help in this instance. This is certainly one of the major topics of our current research.

REFERENCES

[1] I. Adler, N. Karmarkar, M. Resende, and G. Veiga, *Data structures and programming techniques for the implementation of Karmarkar's algorithm*, ORSA J. Comput., 1 (1989), pp. 84–106.

[2] I. Adler and R. Monteiro, *Limiting behavior of the affine scaling continuous trajectories for linear programming problems*, Math. Programming, 50 (1989), pp. 29–51.

[3] I. Adler, M. Resende, G. Veiga, and N. Karmarkar, *An implementation of Karmarkar's algorithm for linear programming*, Math. Programming, 44 (1989), pp. 297–335.

[4] M. Arioli, I. Duff, and D. Ruiz, *Stopping criteria for iterative solvers*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 138–144.

[5] E. Barnes, *A variation on Karmarkar's algorithm for solving linear programming problems*, Math. Programming, 36 (1986), pp. 174–182.

[6] R. Bixby, J. Gregory, I. Lustig, R. Marsten, and D. Shanno, *Very large-scale linear programming: A case study in combining interior point and simplex methods*, Oper. Res., 40 (1992), pp. 885–897.

[7] T. Carpenter, I. Lustig, J. Mulvey, and D. Shanno, *Higher Order Predictor-Corrector Interior Point Methods with Application to Quadratic Objectives*, Rutcor Research Report 67.90, Rutgers University, New Brunswick, NJ, 1990.

[8] I. Dikin, *Iterative solution of problems of linear and quadratic programming*, Soviet Math. Dokl., 8 (1967), pp. 674–675.

[9] P. Gill, W. Murray, and M. Wright, *Practical Optimization*, Academic Press, New York, 1981.

[10] G. Golub and C. V. Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, MD, 1983.

[11] L. Hall and R. Vanderbei, *Two-Third is Sharp for Affine Scaling*, Tech. report, Department of Civil Engineering and Operations Research, Princeton University, Princeton, NJ, 1992.

[12] B. Jansen, C. Roos, T. Terlaky, and J. Vial, *Primal-dual algorithms for linear programming based on the logarithmic barrier method*, J. Optim. Theory Appl., 83 (1994), pp. 1–26.

[13] A. Joshi, S. Goldstein, and P. Vaidya, *A fast implementation of a path-following algorithm for maximizing a linear function over a network polytope*, in Network Flows and Matching: First DIMACS Implementation Challenge, D. Johnson and C. McGeoch, eds., American Mathematical Society, Providence, RI, 1993, pp. 267–298.

[14] J. Kaliski and Y. Ye, *A decomposition variant of the potential reduction algorithm for linear programming*, Management Sci., 39 (1993), pp. 757–776.

[15] N. Karmarkar, *A new polynomial-time algorithm for linear programming*, Combinatorica, 4 (1984), pp. 373–395.

[16] J. Kennington and R. Helgason, *Algorithms for Network Programming*, John Wiley & Sons, New York, 1980.

[17] D. Klingman, A. Napier, and J. Stutz, *Netgen: A program for generating large scale capacitated assignment, transportation and minimum cost flow network problems*, Management Sci., 20 (1974), pp. 814–821.

[18] M. Kojima, S. Mizuno, and A. Yoshise, *A primal-dual interior point algorithm for linear programming*, in Progress in Mathematical Programming, Interior Point and Related Methods, N. Megiddo, ed., Springer-Verlag, New York, 1989, pp. 29–47.

[19] ———, *An $O(\sqrt{n}l)$ iteration potential reduction algorithm for linear complementarity problems*, Math. Programming, 50 (1991), pp. 331–342.

[20] E. Kranich, *SIAG/OPT Views-and-News, a forum for the SIAM Activity Group on Optimization*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.

[21] I. LUSTIG, R. MARSTEN, AND D. SHANNO, *The primal-dual interior point method on the Cray supercomputer*, in Large-Scale Numerical Optimization, SIAM Proc. in Applied Mathematics, Ithaca, NY, 1989, T. Coleman and Y. Li, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1990, pp. 70–80.

[22] ———, *Interior method vs. simplex method: Beyond netlib*, COAL Newsletter, 19 (1991), pp. 41–44.

[23] ———, *On implementing Mehrotra's predictor-corrector interior-point method for linear programming*, SIAM J. Optim., 2 (1992), pp. 435–449.

[24] ———, *Interior methods: Computational state of the art*, ORSA J. Computing, 6 (1994), pp. 1–14.

[25] R. MARSTEN, M. SALTZMAN, D. SHANNO, G. PIERCE, AND J. BALLINTIJN, *Implementation of a dual affine interior point algorithm for linear programming*, ORSA J. Comput., 1 (1989), pp. 287–297.

[26] K. MCSHANE, C. MONMA, AND D. SHANNO, *An implementation of a primal-dual interior-point method for linear programming*, ORSA J. Comput., 1 (1989), pp. 70–83.

[27] S. MEHROTRA, *On the Implementation of a (Primal-Dual) Interior-Point Method*, Tech. report 90-03, Department of Industrial Engineering and Management Science, Northwestern University, Evanston, IL, 1990.

[28] C. MONMA AND A. MORTON, *Computational experience with a dual affine variant of Karmarkar's method for linear programming*, Oper. Res. Lett., 6 (1987), pp. 177–182.

[29] R. MONTEIRO AND I. ADLER, *Interior path following primal-dual algorithms. Part* I: *Linear programming*, Math. Programming, 44 (1989), pp. 27–41.

[30] R. MONTEIRO, I. ADLER, AND M. RESENDE, *A polynomial-time primal-dual affine scaling algorithm for linear and convex quadratic programming and its power series extension*, Math. Oper. Res., 15 (1990), pp. 191–214.

[31] M. MURAMATSU AND T. TSUCHIYA, *A convergence analysis of a long-step variant of the projective scaling algorithm*, Research memorandum, The Institute of Statistical Mathematics, 4–6–7 Minami–Azabu, Minato–Ku, Tokyo 106, Japan, 1992.

[32] G. NEMHAUSER AND L. WOLSEY, *Integer and Combinatorial Optimization*, John Wiley & Sons, New York, 1988.

[33] M. RESENDE, T. TSUCHIYA, AND G. VEIGA, *Identifying the optimal face of a network linear program with a globally convergent interior point method*, in Large Scale Optimization: State of the Art, H. Hager, D. Hearn, and P. Pardalos, eds., Kluwer Academic Publishers, Norwell, MA, 1994, pp. 362–386.

[34] M. RESENDE AND G. VEIGA, *An efficient implementation of a network interior point method*, in Networks Flow and Matching: First DIMACS Implementation Challenge, Vol. 12, D. Johnson and C. McGeoch, eds., American Mathematical Society, Providence, RI, 1993, pp. 299–384.

[35] ———, *An implementation of the dual affine scaling algorithm for minimum-cost flow on bipartite uncapacitated networks*, SIAM J. Optim., 3 (1993), pp. 516–537.

[36] R. VANDERBEI AND J. LAGARIAS, *I. Dikin's convergence result for the affine-scaling algorithm*, in Contemporary Mathematics, Mathematical Developments Arising from Linear Programming, Proc. Joint Summer Research Conference, Bowdoin College, Brunswick, Maine, 1988, J. Lagarias and M. Todd, eds., 1990, pp. 109–119.

[37] R. VANDERBEI, M. MEKETON, AND B. FREEDMAN, *A modification of Karmarkar's linear programming algorithm*, Algorithmica, 1 (1986), pp. 395–407.

[38] Q. YEH, *A Reduced Dual Affine Scaling Algorithm for Solving Assignment and Transportation Problems*, Ph.D. thesis, Columbia University, New York, NY, 1989.

SIAM J. SCI. COMPUT.
Vol. 17, No. 5, pp. 1224–1231, September 1996

012

COMPUTATION OF THE NONCENTRAL GAMMA DISTRIBUTION*

L. KNÜSEL† AND B. BABLOK†

Abstract. This paper deals with the computation of upper and lower tail probabilities of the noncentral gamma distribution and with the computation of the noncentrality parameter given the upper or lower tail probability. The tail probabilities are computed in an efficient and numerically stable manner with a given relative accuracy even for small probabilities and for a wide range of parameters where other algorithms found in the statistical literature fail. The basic ideas in this paper can also be applied to the noncentral chisquare, beta, F, and t distributions.

Key words. noncentral distributions, noncentrality parameter

AMS subject classification. 65U05

1. Overview.

Noncentral distributions play an important role in statistics, e.g., when we are interested in the power function of the chisquare, F, or t test. In §2 we give the definition of the central and noncentral gamma and beta distributions and indicate how the (central and noncentral) chisquare, F, and t distributions can be derived from these distributions which are mathematically and numerically simpler to handle. In §3 we give the basic ideas of the computation of the central gamma distribution. The algorithms are based upon open forward and backward recursions as described in Knüsel [5]. Section 4 deals with the computation of the noncentral gamma distribution and §5 shows how we can compute the noncentrality parameter given the other parameters and the upper or lower tail probability of the distribution by using the Newton algorithms for finding zeros of a real function.[1] The essential point is the fact that the derivative with respect to λ of the noncentral gamma distribution can also be computed in a numerically stable and fast way with a given relative precision using ideas similar to those in §§3 and 4. We want to point out that for many applications the computation of the noncentrality parameter is of higher practical relevance than the computation of quantiles (e.g., for the computation of the effect size or the necessary sample size at a given power).

We can compute the noncentral beta distribution in a manner analogous to that of the gamma distribution. So, all three commonly used noncentral distributions (chisquare, F, and t) can be computed by the methods described in this paper as these distributions can be derived from the gamma and beta distributions.

2. Relationship between the different distributions.

In this section we want to give in short the definition of the gamma and beta distributions considered and to show the relationship between these distributions and the classical central and noncentral chisquare, F, and t distributions.

Central gamma distribution. Let X denote a random variable with a (central) gamma distribution with parameter $a > 0$. The distribution function of X is defined as

$$F(x|a) = Pr\{X < x\} = \frac{1}{\Gamma(a)} \int_0^x t^{a-1} e^{-t} dt \quad \text{for} \quad x > 0,$$

where

$$\Gamma(a) = \int_0^\infty t^{a-1} e^{-t} dt.$$

*Received by the editors February 23, 1994; accepted for publication (in revised form) May 22, 1995.

†Department of Statistics, University of Munich, 80539 Munich, Germany (leo.knuesel@lrz.uni-muenchen.de; deccpfl9@ibmmail.com).

[1] The ideas in these two sections are mainly due to Bablok (cf. Bablok [1]).

The function $\Gamma(a)$ is the (complete) gamma function while the integral in the definition of $F(x|a)$ is called the incomplete gamma function.

Central beta distribution. Let X_1 and X_2 be independent random variables having gamma distributions with parameters a and b, respectively, and let $X = X_1/(X_1 + X_2)$. Then X has a beta distribution with parameters a and b and the distribution function of X is given by

$$F(x|a,b) = Pr\{X < x\} = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \int_0^x t^{a-1}(1-t)^{b-1}dt \quad \text{for} \quad 0 < x < 1.$$

The function $\Gamma(a)\Gamma(b)/\Gamma(a+b)$ is called the (complete) beta function while the integral in the definition of $F(x|a,b)$ is called the incomplete beta function.

Noncentral gamma distribution. Let X denote a random variable with a noncentral gamma distribution with parameter $a > 0$ and with noncentrality parameter $\lambda > 0$. The distribution function of X is defined as

$$F(x|a,\lambda) = Pr\{X < x\} = \sum_{i=0}^{\infty} p_\lambda(i)F(x|a+i), \qquad \text{where} \qquad p_\lambda(i) = \frac{e^{-\lambda}\lambda^i}{i!}$$

(see Patnaik [7]). So, the noncentral gamma distribution is a mixture of central gamma distributions $F(x|a+i)$ with Poisson weights $p_\lambda(i)$.

Noncentral beta distribution. Let X_1 and X_2 be independent random variables: X_1 with a noncentral gamma distribution with parameter $a > 0$ and noncentrality parameter $\lambda > 0$, and X_2 with a central gamma distribution with parameter $b > 0$. Then the random variable $X = X_1/(X_1 + X_2)$ has a noncentral beta distribution with parameters a and b and with noncentrality parameter λ, and the distribution function of X is given by

$$F(x|a,b,\lambda) = Pr\{X < x\} = \sum_{i=0}^{\infty} p_\lambda(i)F(x|a+i,b), \qquad \text{where} \qquad p_\lambda(i) = \frac{e^{-\lambda}\lambda^i}{i!}$$

(see Patnaik [7]). So, the noncentral beta distribution is a mixture of central beta distributions $F(x|a+i,b)$ with Poisson weights $p_\lambda(i)$.

The following results concerning the chisquare, F, and t distributions can also be derived from Patnaik [7] or they can be found in Johnson–Kotz–Balakrishnan [4]. If X has a noncentral chisquare distribution with noncentrality parameter λ, then the random variable $Y = X/2$ has a noncentral gamma distribution with parameter $a = n/2$ and with noncentrality parameter $\lambda' = \lambda/2$. If X has a noncentral F distribution with (n_1, n_2) degrees of freedom and with noncentrality parameter λ, then the random variable $Y = n_1X/(n_2+n_1X)$ has a noncentral beta distribution with parameters $a = n_1/2, b = n_2/2$ and with noncentrality parameter $\lambda' = \lambda/2$. If X has a noncentral t distribution with n degrees of freedom and with noncentrality parameter δ, then the random variable $Y = X^2$ has a noncentral F distribution with $(1, n)$ degrees of freedom and with noncentrality parameter $\lambda = \delta^2$. As the noncentral t distribution is no longer symmetric, only probabilities of the form $Pr\{|X| < t\}$ or $Pr\{|X| > t\}$ can be computed by means of the F or beta distribution. Probabilities of the form $Pr\{X < t\}$ or $Pr\{X > t\}$ cannot be found in this way.

So, all three commonly used central and noncentral distributions (chisquare, F, and t) can be derived from the central and noncentral gamma and beta distributions.

3. Computation of the (central) gamma distribution. This section is a short summary of the methods used to compute the upper and lower tail probability of the central gamma distribution as described in more detail in Knüsel [5]. Let X denote a random variable having a (central) gamma distribution with parameter $a > 0$:

$$(1) \qquad Pr\{X < x\} = \frac{1}{\Gamma(a)} \int_0^x t^{a-1}e^{-t}dt \quad \text{for} \quad x > 0,$$

where $\Gamma(a)$ denotes the gamma function

$$\Gamma(a) = \int_0^\infty t^{a-1}e^{-t}dt. \tag{2}$$

We define the following three basic quantities for $a > 0, x > 0$:

$$\begin{aligned} p(a,x) &= \frac{e^{-x}x^{a-1}}{\Gamma(a)}, \\ I(a,x) &= xe^{-x}\int_0^1 t^{a-1}e^{-xt}dt, \\ J(a,x) &= xe^{-x}\int_1^\infty t^{a-1}e^{-xt}dt. \end{aligned} \tag{3}$$

Denoting by $F(x|a)$ and $F^c(x|a)$ the distribution function and the complementary distribution function of X, respectively, we have

$$\begin{aligned} F(x|a) &= Pr\{X < x\} = p(a,x)I(a,x), \\ F^c(x|a) &= Pr\{X > x\} = p(a,x)J(a,x). \end{aligned} \tag{4}$$

The two probabilities add up to one, and therefore it is sufficient to compute the smaller of the two terms. As the median of the gamma distribution is about a if a is not too small, we therefore have, as a rough rule, to compute $I(a,x)$ when $x \le a$ and $J(a,x)$ when $x > a$. From the definition of $I_a = I(a,x)$ and $J_a = J(a,x)$, we derive the following properties for any given $x > 0$:

$$\begin{aligned} &I_a \downarrow 0 \quad \text{for} \quad a \uparrow \infty, \\ &I_a = \frac{x}{a}(1 + I_{a+1}) \quad \text{for} \quad a > 0 \end{aligned} \tag{5}$$

and

$$\begin{aligned} &J_a \downarrow 0 \quad \text{for} \quad a \downarrow -\infty \quad (J_1 = 1), \\ &J_{a+1} = \left(1 + \frac{a}{x}J_a\right) \quad \text{for} \quad a \in R. \end{aligned} \tag{6}$$

Using these properties, we can compute the quantities I_a and J_a with open recursions without knowing an exact starting value but with the approximate starting value zero. In the case of I_a, the following open backward recursion can be used:

Set $\tilde{I}_{a+n} = 0$ for some positive integer n.
For $b = a + n - 1, a + n - 2, \dots, a$ compute

$$\tilde{I}_b = \frac{x}{b}(1 + \tilde{I}_{b+1}). \tag{7}$$

Then $\tilde{I}_a$ is an approximation to $I_a = I(a,x)$ with the relative error

$$0 \le \frac{I_a - \tilde{I}_a}{I_a} \le fac, \quad \text{where} \quad fac = \frac{x^n}{a(a+1)\cdots(a+n-1)}.$$

Therefore, we can guarantee a relative error of less than a given $\varepsilon > 0$ if we choose n as the smallest positive integer such that$fac < \varepsilon$. In regard to the computation of J_a, we can use the following forward recursion:

Set $\tilde{J}_{a-n} = 0$ for some positive integer $n \leq a$.
For $b = a - n, a - n + 1, \dots, a - 1$ compute

$$\tilde{J}_{b+1} = 1 + \frac{b}{x}\tilde{J}_b. \tag{8}$$

Then $\tilde{J}_a$ is an approximation to $J_a = J(a, x)$ with the relative error

$$0 \leq \frac{J_a - \tilde{J}_a}{J_a} \leq fac, \quad \text{where} \quad fac = \frac{(a-1)(a-2)\cdots(a-n)}{x^n}.$$

Note that $\tilde{J}_{a-n} = 0$ implies $\tilde{J}_{a-n+1} = 1$ and so the first step in the recursion could be saved. Thus, we can guarantee a relative error of less than a given $\varepsilon > 0$ if we choose n as the smallest positive integer such that $fac < \varepsilon$. If such an integer does not exist, we have to compute the exact starting value J_b for $0 < b < 1$ for our recursion.

4. Computation of the noncentral gamma distribution. Now, let X denote a random variable having a noncentral gamma distribution with parameter a and noncentrality parameter λ. The distribution function of X can be written as a mixture of central gamma distributions with Poisson weights:

$$\begin{aligned} F(x|a,\lambda) &= Pr\{X < x\} = \sum_{i=0}^{\infty} p_\lambda(i) F(x|a+i), \\ F^c(x|a,\lambda) &= Pr\{X > x\} = \sum_{i=0}^{\infty} p_\lambda(i) F^c(x|a+i) \end{aligned} \tag{9}$$

with

$$p_\lambda(i) = \frac{e^{-\lambda}\lambda^i}{i!} = Pr\{Po(\lambda) = i\},$$

where $Po(\lambda)$ denotes a random variable with a Poisson distribution with parameter λ (cf. Patnaik [7]). Note, that $p_\lambda(i)$ is related to the function $p(a, x)$ defined in (3) by $p_\lambda(i) = p(i + 1, \lambda)$. We want to show how both probabilities in (9) can be computed with a given relative error by using techniques similar to those used for the central distributions. First, we deal with the computation of the lower tail probability $F(x|a, \lambda)$ for given x, a, and λ. For the central gamma distribution we have the relations (4) and (5). Furthermore, not only the component $I_a = I(a, x)$ but also the distribution function $F(x|a) = p(a, x)I(a, x)$ itself is monotonous in a:

$$F(x|a) \downarrow 0 \quad \text{for} \quad a \uparrow \infty. \tag{10}$$

This follows from the addition theorem of the gamma distribution; the sum of two gamma variables has again a gamma distribution, the parameter being the sum of the individual parameters. Now, we decompose the lower tail probability $F(x|a, \lambda)$ in three terms:

$$F(x|a,\lambda) = \sum_{i=0}^{k_1} \cdots + \sum_{i=k_1+1}^{k_2-1} \cdots + \sum_{i=k_2}^{\infty} \cdots = S_1 + S_2 + S_3. \tag{11}$$

The number k_2 is chosen as the smallest integer such that

$$Pr\{Po(\lambda) \geq k_2\} \leq \varepsilon, \tag{12}$$

i.e., k_2 is the upper ε-quantile of a Poisson distribution with parameter λ. Because of the monotonicity relation (10) we have

$$\begin{aligned} S_3 &\le F(x|a+k_2)Pr\{Po(\lambda) \ge k_2\} \le F(x|a+k_2)\varepsilon, \\ S_1 + S_2 &\ge F(x|a+k_2)Pr\{Po(\lambda) < k_2\} \ge F(x|a+k_2)(1-\varepsilon), \end{aligned} \tag{13}$$

and so

$$\frac{S_3}{S_1+S_2} \le \frac{\varepsilon}{1-\varepsilon} \approx \varepsilon, \quad \text{i.e.,} \quad S_3 \le \varepsilon(S_1+S_2) \quad \text{(approximately).} \tag{14}$$

The sum S_2 can be written as

$$S_2 = \sum_{i=k_1+1}^{k_2-1} T_i, \tag{15}$$

where $T_i = T_i(x,a,\lambda) = p_\lambda(i)F(x|a+i) = p(i+1,\lambda)p(a+i,x)I(a+i,x)$. Using the obvious relation

$$p(a+1,x) = \frac{x}{a}p(a,x), \tag{16}$$

we find

$$T_i = q_i T_{i+1}, \quad \text{where} \quad q_i = \frac{i+1}{\lambda}\frac{a+i}{x}\left(\frac{I(a+i,x)}{I(a+i+1,x)}\right). \tag{17}$$

Starting the summation of S_2 at the end $i = k_2 - 1$ and using the backward recursion (5) for $I_{a+i} = I(a+i,x)$, we can compute each term T_i of this sum with a relative error smaller than ε and thus, all terms being positive, the overall relative error of $\tilde{S}_2$, the computed value of S_2, is also smaller than ε. We must now determine the value k_1 such that $S_1 \le \varepsilon S_2$. Using the backward recursion (5) for $I_{a+i} = I(a+i,x)$ in (17), we obtain

$$q_i = \frac{i+1}{\lambda}\left(\frac{1}{I(a+i+1,x)} + 1\right), \tag{18}$$

and due to the monotonicity property (5) of $I_a = I(a,x)$ it follows that

$$q_i = T_i/T_{i+1} \quad \text{is decreasing for} \quad i \downarrow 0. \tag{19}$$

This means that the terms $T_i (i = k_2 - 1, k_2 - 2, \dots, 0)$ in the sum $S_1 + S_2$ are increasing as long as $q_i > 1$ and decreasing as soon as $q_i < 1$ and that the partial sum S_1 can be estimated by the geometric series

$$S_1 = \sum_{i=0}^{k_1} T_i \le \frac{T_{k_1}}{1-q_{k_1}}, \quad \text{where} \quad q_{k_1} = \frac{T_{k_1}}{T_{k_1+1}} \quad (\text{if } q_{k_1} < 1). \tag{20}$$

Now, if we stop the summation of S_2 as soon as

$$\frac{T_{k_1}}{1-q_{k_1}} \le \varepsilon \tilde{S}_2, \tag{21}$$

then the approximation $F(x|a,\lambda) \approx \tilde{S}_2$ has a relative error smaller than 3ε (approximately) as $S_1 \le \varepsilon\tilde{S}_2$ due to (20) and (21), $S_3 \le \varepsilon S_2$ due to (14), and because the relative error of $\tilde{S}_2$, the computed value of S_2, is also smaller than ε.

We want to point out that it is crucial for the computation of S_2 to start off the summation at the upper end k_2. Many algorithms found in the literature (see, e.g., Frick [3], Lenth [6], Farebrother [2]) start the summation at the lower ε-quantile or even at zero and proceed toward the upper end using a recursion similar to (7) but in the forward instead of the backward direction. This summation is instable due to numerical cancellation and the relative error cannot be kept under control. The paper of Wang and Gray [8] uses approximation methods without known approximation bounds.

The computation of the upper tail probability $F^c(x|a,\lambda)$ can be performed in quite an analogous way. The series (9) is decomposed in three terms:

$$F^c(x|a,\lambda) = \sum_{i=0}^{k_1} \cdots + \sum_{i=k_1+1}^{k_2-1} \cdots + \sum_{i=k_2}^{\infty} \cdots = S_1 + S_2 + S_3. \tag{22}$$

This time we have to use the recurrence relation (6) for $J(a,x)$ that works in the forward direction. The number k_1 is chosen as the largest integer such that

$$Pr\{Po(\lambda) \le k_1\} \le \varepsilon. \tag{23}$$

Because of the monotonicity relation

$$F^c(x|a) \downarrow 0 \quad \text{for} \quad a \downarrow 0 \quad \text{and} \quad x > 0, \tag{24}$$

we have estimations completely analogous to those of the lower tail probability and further computations offer no new problems.

5. Computation of the noncentrality parameter. In this section we want to compute the noncentrality parameter λ of a noncentral gamma distribution given the parameters a and x and given the lower tail probability p:

(25) For given x, a, and p determine λ such that $F(x|a,\lambda) = p$ with $p \le \frac{1}{2}$.

If the upper tail probability were given we would use the complementary distribution function $F^c(x|a,\lambda)$ instead of $F(x|a,\lambda)$ to avoid cancellation. We want to point out that for noncentral distributions the inversion with respect to the noncentrality parameter is of higher practical relevance than the computation of classical quantiles in many applications (e.g., for the computation of the effect size or the necessary sample size at a given power). Now we want to show how to solve the problem (25). For given x and a we have $F(x|a,\lambda) \downarrow 0$ for $\lambda \uparrow \infty$ as we shall see in a moment, and therefore the probability p must be smaller than $F(x|a,0) = F(x|a)$ or there exists no solution to (25). We propose using the Newton algorithm for finding zeros of a function $f(\lambda)$:

$$\lambda_{i+1} = \lambda_i - \frac{f(\lambda_i)}{f'(\lambda_i)}. \tag{26}$$

In our case we have $f(\lambda) = F(x|a,\lambda) - p$ with $x > 0, a > 0, 0 < p \le F(x|a)$, and $\lambda \ge 0$. The computation of $f(\lambda)$ can be performed using the methods in the previous sections and all we need is a numerically stable algorithm for the computation of $f'(\lambda)$ that can guarantee a given relative precision. Relative precision is crucial as otherwise the recursion becomes useless if $f'(\lambda)$ becomes small, although a moderate relative precision of 10^{-3} or so would be sufficient. From the representation (9) we derive

$$f'(\lambda) = \frac{d}{d\lambda}F(x|a,\lambda) = \sum_{i=0}^{\infty} \frac{d}{d\lambda}p(i+1,\lambda)F(x,a+i), \tag{27}$$

and as

$$\frac{d}{d\lambda}p(i+1,\lambda) = \begin{cases} -p(i+1,\lambda) + p(i,\lambda) & \text{for} \quad i > 0, \\ -p(i+1,\lambda) & \text{for} \quad i = 0, \end{cases} \tag{28}$$

we find

$$f'(\lambda) = -F(x|a,\lambda) + F(x|a+1,\lambda). \tag{29}$$

As $f'(\lambda)$ is the first derivative of $F(x|a,\lambda)$ with respect to λ, we could also compute derivatives of higher order by iterative application of (29). Now, the monotonicity property (10) for the central gamma distribution is also valid for noncentral distributions:

$$F(x|a,\lambda) \downarrow 0 \quad \text{for} \quad a \uparrow \infty. \tag{30}$$

This is obvious from the representation (9) of the noncentral distribution. So, we see from (29) that $f'(\lambda) < 0$ and this proves the above-mentioned monotonicity property in λ. Now, using the factorization $F(x|a) = p(a,x)I(a,x)$ given by (4) and the recurrence relations (5) and (16) for the factors $I(a,x)$ and $p(a,x)$, equation (29) simplifies to

$$f'(\lambda) = -\sum_{i=0}^{\infty} t_i, \quad \text{where} \quad t_i = t_i(x,a,\lambda) = p(i+1,\lambda)p(a+i+1,x). \tag{31}$$

Again using the recurrence relation (16) for $p(a,x)$, we find

$$t_i = c_i t_{i-1} \quad \text{with} \quad c_i = \frac{\lambda x}{i(a+i)} \quad \text{for} \quad i > 0. \tag{32}$$

We see that $c_i \downarrow 0$ for $i \uparrow \infty$ which means that for increasing i the terms t_i are increasing as long as $c_i > 1$ and decreasing as soon as $c_i < 1$. Let k be the smallest integer i with $c_i < 1$, i.e.,

$$k = \lceil \sqrt{a^2/4 + \lambda x} - a/2 \rceil. \tag{33}$$

Then the term t_k is the largest term in the series $\sum t_i$. Now, we decompose this series as follows:

$$f'(\lambda) = -\sum_{i=0}^{\infty} t_i = -\left(\sum_{i=0}^{k-1} t_i + t_k + \sum_{i=k+1}^{\infty} t_i \right) = -(L_k + t_k + U_k). \tag{34}$$

We have

$$\begin{aligned} U_k &= t_{k+1} + t_{k+2} + \cdots \\ &= t_k(c_{k+1} + c_{k+1}c_{k+2} + \cdots) \\ &= t_k I_k, \end{aligned} \tag{35}$$

and from this we see that

$$\begin{aligned} &I_k \downarrow 0 \quad \text{for} \quad k \uparrow \infty, \\ &I_{k-1} = c_k(1 + I_k). \end{aligned} \tag{36}$$

As (36) is equivalent to (5), we can compute the factor I_k with a given relative error ε in exactly the same way as in (7). For the term L_k we use the recurrence relation (32) in the other direction:

$$t_i = d_i t_{i+1} \quad \text{with} \quad d_i = \frac{(i+1)(a+i+1)}{\lambda x} = \frac{1}{c_{i+1}} \quad \text{for} \quad i \geq 0 \tag{37}$$

and now we can write

$$\begin{aligned} L_k &= t_{k-1} + t_{k-2} + \cdots \\ &= t_k(d_{k-1} + d_{k-1}d_{k-2} + \cdots) \qquad (38) \\ &= t_k J_k. \end{aligned}$$

From this we see that

$$\begin{aligned} J_k \downarrow 0 \quad &\text{for} \quad k \downarrow 0, \\ J_{k+1} &= d_k(1 + J_k). \end{aligned} \qquad (39)$$

Note that this is completely symmetric to (36) up to the direction of the recursion, and J_k can be computed in exactly the same way as the quantity I_k up to the direction of the recursion. Also notice that $J_0 = 0$ and so the open recursion with the approximate starting value $\tilde{J}_{k-n} = 0$ becomes exact if $n = k$; the problem with the exact starting value J_b for $0 < b < 1$ in the recursion (8) does not exist here as k is an integer. If we set $\bar{L}_k = L_k + t_k$ and $\bar{J}_k = J_k + 1$ then we have the recurrence relation $\bar{J}_{k+1} = 1 + d_k \bar{J}_k$ which is equivalent to (6). But as the number of steps to compute the quantities J_k and $\bar{J}_k$ is exactly the same (up to the first trivial step when computing $\bar{J}_k$), we prefer here the quantity J_k that is completely symmetric to I_k. So, all terms t_k, I_k, and J_k can be computed with a given relative precision and this means that

$$f'(\lambda) = -\sum_{i=0}^{\infty} t_i = -(L_k + t_k + U_k) = -t_k(J_k + 1 + I_k) \qquad (40)$$

can also be computed with a given relative precision.

So, the numerator $f(\lambda_i)$ and the denominator $f'(\lambda_i)$ in the Newton algorithm (26) can be computed with the methods described in this paper. Further, a useful starting value can be derived from the facts that the distribution of a standardized noncentral gamma variable is approximately normal if a is not too small and the expectation and the variance of a noncentral gamma variable are $a + \lambda$ and $a + 2\lambda$, respectively.

REFERENCES

[1] B. Bablok, *Numerische Berechnung nicht-zentraler statistischer Verteilungen*, Diploma thesis, Department of Statistics, University of Munich, Munich, Germany, 1988.

[2] R. W. Farebrother, *The distribution of a noncentral chisquare variable with non-negative degrees of freedom*, Appl. Statist., 36 (1987), pp. 402–405.

[3] H. Frick, *A remark on algorithm AS* 226: *Computing non-central beta probabilities*, Appl. Statist., 39 (1990), pp. 311-312.

[4] N. L. Johnson, S. Kotz, and N. Balakrishnan, *Continuous Univariate Distributions*, Vol. 2, Second Ed., John Wiley, New York, 1995.

[5] L. Knüsel, *Computation of the chisquare and Poisson distribution*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 1022–1036.

[6] R. V. Lenth, *Algorithm AS* 226: *Computing noncentral beta probabilities*, Appl. Statist., 36 (1987), pp. 241–244.

[7] P. B. Patnaik, *The noncentral chi-square and F-distributions and their applications*, Biometrika, 36 (1949), pp. 202–232.

[8] S. Wang and H. L. Gray, *Approximating tail probabilities of noncentral distributions*, Comput. Statist. Data Anal., 15 (1993), pp. 343–352.

SIAM J. SCI. COMPUT.
Vol. 17, No. 5, pp. 1232–1248, September 1996

013

LOCATING AND COMPUTING ALL THE SIMPLE ROOTS AND EXTREMA OF A FUNCTION*

DIMITRIS J. KAVVADIAS†‡ AND MICHAEL N. VRAHATIS†

Abstract. This paper describes and analyzes two algorithms for locating and computing with certainty all the simple roots of a twice continuously differentiable function $f: (a, b) \subset \mathbb{R} \to \mathbb{R}$ and all the extrema of a three times continuously differentiable function in (a, b). The first algorithm locates and computes all the simple roots or all the extrema, while the second one is more efficient in the case where both simple roots and extrema are required.

This paper also gives analytical estimation of the expected complexity of the algorithms based on the distribution of the roots in (a, b). Here only the case of uniform distribution is examined, which is also the approach to be followed when no statistical data are available for the function at hand.

The algorithms have been implemented and tested. Performance information for a well-known Bessel function is reported.

Key words. zeros isolation, Kronecker–Picard theory, topological degree, locating simple roots and extrema, computing simple roots and extrema, combinatorial optimization, zeros identifications, distribution of the roots and extrema, expected algorithms complexity, Bessel functions

AMS subject classifications. 65H05, 68C25

1. Introduction.

It is well known that information concerning all the roots and/or all the extrema of a function

$$f: (a, b) \subset \mathbb{R} \to \mathbb{R} \tag{1.1}$$

is of major importance in many different fields of science and technology.

An immediate method to accomplish this task suggests scanning the interval (a, b) and applying some rootfinding method for each consecutive root. The method that is employed here is heavily based on the knowledge of the total number of roots within (a, b). In order to obtain this information we use results from topological degree theory and especially from the theory of the Kronecker–Picard integral [24, 25]. This theory gives a formula for the computation of the total number of roots of a system of equations within a given region. With this tool in hand one can construct a procedure for the localization and isolation of all the roots by dividing the given region successively and applying the above formula to these subregions until the final domains contain at most one root. To our knowledge, the first method to this end was introduced by Hoenders and Slump [10, 11, 28], who recently reconsidered and applied Kronecker–Picard theory to calculate the total number of simple roots of a system of nonlinear equations as well as to calculate the total number of multiple roots of a single equation of any multiplicity.

Other approaches that have been used successfully to find all solutions of systems of equations as well as the global optimum of a function are based on interval analysis (see, for example, [1, 7, 8, 9, 15, 16, 17]). The corresponding existence tool of these methods is the availability of the range of the function in a given interval, which can be implemented very efficiently using interval arithmetic, though accuracy problems must be resolved. This tool, however, reports with certainty only the negative case, i.e., when no roots are in the interval. The positive case proceeds by subdividing the interval into two halves and employing additional criteria. This case is common when the function has many extrema but few roots in the interval of interest. In addition, this tool provides the existence of a root and not their exact number, which means that more sophisticated decisions (than merely subdividing) cannot be made.

*Received by the editors March 30, 1994; accepted for publication (in revised form) May 15, 1995. This work was partially supported by the EEC ESPRIT Basic Research Action 7141 (ALCOM II).

†Department of Mathematics, University of Patras, GR-261.10 Patras, Greece (vrahatis@math.upatras.gr).

‡Computer Technology Institute, P.O. Box 1122, GR-261.10 Patras, Greece (djk@math.upatras.gr).

In this paper we implement Kronecker–Picard theory and give a method for locating and computing all the simple roots of a twice continuously differentiable function and all the extrema of a three times continuously differentiable function in an interval (a, b). In addition, we give analytical expressions of the total *expected* work needed in achieving the isolation and computation of all the roots of (1.1) under certain assumptions that we present in the sequel. This may prove to be very useful in real life applications since very early estimations of the total running time of the algorithm are available, contrary to other methods which are totally unpredictable. Nevertheless, it is important to notice that our analytical results and techniques also apply to any method that proceeds by repeated subdivisions, giving the expected depth of the subdivision under the same assumptions. Moreover, the main computational burden of our method comes from the need of an integration, a problem that has been studied extensively and for which numerous methods exist.

The rootfinding portion of our method employs a modification of the well-known bisection method. Alternatively, any one of the one-dimensional rootfinding methods (see [23, 20, 21, 26, 2]) can be used. We use the bisection method for several reasons which we explain in §2.2, among which is its known behavior concerning the number of iterations required when we seek the root with a predetermined accuracy.

Here we study the one-dimensional case only. The reason for this restriction is that the corresponding analysis of our method to n dimensions seems to be quite different.

2. Theoretical aspects. Our algorithms are separated in two phases: (1) the phase of the localization and the isolation of all the roots, and (2) the rootfinding phase for the computation of all the roots within a predetermined accuracy.

For the localization phase we use a method of determining the total number of simple roots of a single equation within a given interval. To this end we briefly exploit degree theory for determining the exact number of roots of a single equation by computing the value of topological degree using Kronecker's integral [19, 3, 29, 22, 13] on Picard's extension [24, 25, 10, 28, 11].

For the rootfinding phase we have chosen the bisection method for reasons to be explained later.

2.1. The topological degree for the computation of the total number of roots and extrema. Let us first define the notion of the topological degree. Suppose that the function $F_n = (f_1, \dots, f_n): \mathcal{D}_n \subset \mathbb{R}^n \to \mathbb{R}^n$ is defined and two times continuously differentiable in a bounded domain $\mathcal{D}_n$ of $\mathbb{R}^n$ with boundary $b(\mathcal{D}_n)$. Suppose further that the roots of the equation $F_n(x) = \Theta_n$ ($\Theta_n = (0, \dots, 0)$ denotes the origin of $\mathbb{R}^n$) are not located on $b(\mathcal{D}_n)$ and they are simple; i.e., the Jacobian determinant of F_n at these roots is nonzero. Then the *topological degree of F_n at Θ_n relative to $\mathcal{D}_n$* is denoted by $\deg[F_n, \mathcal{D}_n, \Theta_n]$ and can be defined by the following sum:

$$\deg[F_n, \mathcal{D}_n, \Theta_n] = \sum_{x \in F_n^{-1}(\Theta_n)} \operatorname{sgn} J_{F_n}(x), \tag{2.1}$$

where J_{F_n} denotes the determinant of the Jacobian matrix and sgn defines the sign function.

The above definition can be generalized when F_n is only continuous [23]. In this case, Kronecker's theorem [3, 23] states that $F_n(x) = \Theta_n$ has at least one root in $\mathcal{D}_n$ if $\deg[F_n, \mathcal{D}_n, \Theta_n] \neq 0$. Furthermore, if $\mathcal{D}_n = \mathcal{D}_n^1 \cup \mathcal{D}_n^2$ where $\mathcal{D}_n^1$ and $\mathcal{D}_n^2$ have disjoint interiors and $F_n(x) \neq \Theta_n$ for all $x \in b(\mathcal{D}_n^1) \cup b(\mathcal{D}_n^2)$, then the topological degree is additive.

Several methods for the computation of the topological degree have been proposed in the past few years [29, 22, 13, 14, 30, 31, 4]. Also, $\deg[F_n, \mathcal{D}_n, \Theta_n]$ can be represented by the

Kronecker integral, which is closely tied with facts used later and is defined as follows:

$$(2.2)\quad \deg[F_n, \mathcal{D}_n, \Theta_n] = \frac{\Gamma(n/2)}{2\pi^{n/2}} \int\int_{b(\mathcal{D}_n)} \cdots \int \frac{\sum_{i=1}^{n} A_i dx_1 \ldots dx_{i-1} dx_{i+1} \ldots dx_n}{({f_1}^2 + {f_2}^2 + \ldots + {f_n}^2)^{n/2}},$$

where A_i is defined by the following determinant:

$$(2.3)\qquad A_i = (-1)^{n(i-1)} \begin{vmatrix} F_n & \dfrac{\partial F_n}{\partial x_1} & \cdots & \dfrac{\partial F_n}{\partial x_{i-1}} & \dfrac{\partial F_n}{\partial x_{i+1}} & \cdots & \dfrac{\partial F_n}{\partial x_n} \end{vmatrix}.$$

Now, since $\deg[F_n, \mathcal{D}_n, \Theta_n]$ is equal to the number of simple roots of $F_n(x) = \Theta_n$ which give positive Jacobian minus the number of simple roots which give negative Jacobian, then of course the total number $\mathcal{N}^r$ of simple roots of $F_n(x) = \Theta_n$ can be obtained by the value of $\deg[F_n, \mathcal{D}_n, \Theta_n]$ if all these roots give the same Jacobian sign. To this end Picard considered the following extensions of the function F_n and the domain $\mathcal{D}_n$:

$$(2.4)\qquad F_{n+1} = (f_1, \ldots, f_n, f_{n+1}) \colon \mathcal{D}_{n+1} \subset \mathbb{R}^{n+1} \to \mathbb{R}^{n+1},$$

where $f_{n+1} = y\, J_{F_n}$, $\mathbb{R}^{n+1} : x_1, x_2, \ldots, x_n, y$, and $\mathcal{D}_{n+1}$ is the direct product of the domain $\mathcal{D}_n$ with an arbitrary interval of the real y–axis containing the point $y = 0$. Then the roots of the system of equations

$$(2.5)\qquad \begin{aligned} &f_i(x_1, x_2, \ldots, x_n) = 0, \quad i = 1, \ldots, n, \\ &y\, J_{F_n}(x_1, x_2, \ldots, x_n) = 0 \end{aligned}$$

are the same simple roots of $F_n(x) = \Theta_n$ provided $y = 0$. On the other hand, it is easily seen that the Jacobian of (2.5) is equal to $\left(J_{F_n}(x)\right)^2$, which is always positive. Thus, we conclude that the total number $\mathcal{N}^r$ of roots of $F_n(x) = \Theta_n$ can be given by the following relation:

$$(2.6)\qquad \mathcal{N}^r = \deg[F_{n+1}, \mathcal{D}_{n+1}, \Theta_{n+1}].$$

We consider now the problem of calculating the total number of simple roots of $f(x) = 0$, where $f \colon (a, b) \subset \mathbb{R} \to \mathbb{R}$ is twice continuously differentiable in a predetermined interval (a, b), where a and b are arbitrarily chosen such that $f(a)\, f(b) \neq 0$. According to Picard's extension we define the function $F_2 = (f_1, f_2) \colon \mathcal{P} \subset \mathbb{R}^2 \to \mathbb{R}^2$ and the corresponding system

$$(2.7)\qquad \begin{aligned} f_1(x, y) &= f(x) = 0, \\ f_2(x, y) &= yf'(x) = 0, \end{aligned}$$

where the prime denotes differentiation and where $\mathcal{P}$ is an arbitrarily chosen rectangular parallelepiped in the (x, y)–plane given by $a \leq x \leq b$ and $-\gamma \leq y \leq \gamma$ with γ a small positive constant.

Now, since the roots are simple, which means $f'(x) \neq 0$ for $x \in f^{-1}(0)$, it is easily seen that the roots of (2.7) in the above-defined region are the same roots as (1.1). Also, since $J_{F_2} = {f'}^2$, the total number of simple zeros $\mathcal{N}^r$ of the function (1.1) in (a, b) can be given by

$$(2.8)\qquad \mathcal{N}^r = \deg[F_2, \mathcal{P}, \Theta_2].$$

Now, for $n = 2$, by applying (2.1) and using the relations $df_j = \frac{\partial f_j}{\partial x_1} dx_1 + \frac{\partial f_j}{\partial x_2} dx_2$, where $j = 1, 2$, we can easily obtain

$$(2.9)\qquad \mathcal{N}^r = \frac{1}{2\pi} \oint_{b(\mathcal{P})} \frac{f_1 df_2 - f_2 df_1}{{f_1}^2 + {f_2}^2} = \frac{1}{2\pi} \oint_{b(\mathcal{P})} d \arctan\left(\frac{f_2}{f_1}\right).$$

Replacing f_1 and f_2 by virtue of (2.7) and performing the integration in (2.9) we finally get

$$\mathcal{N}^r = -\frac{1}{\pi}\left[\gamma \int_a^b \frac{f(x)f''(x) - f'^2(x)}{f^2(x) + \gamma^2 f'^2(x)}\, dx + \arctan\left(\frac{\gamma f'(b)}{f(b)}\right) - \arctan\left(\frac{\gamma f'(a)}{f(a)}\right)\right]. \tag{2.10}$$

It has been shown by Picard [24, 25] that the relation (2.10) is independent of the value of γ.

Of course the total number $\mathcal{N}^e$ of the extrema of $f \in C^3$, i.e., $x \in (a, b)$ such that $f'(x) = 0$, can be obtained by setting in (2.9) $f_1 = f'$ and $f_2 = yf''$.

Remark 2.1. Using (2.10) it is possible that in many applications $\mathcal{N}^r$ can be computed analytically. If not, one can use numerical integration [26].

The Kronecker–Picard integral can also be applied for the determination of the total number of multiple roots [5, 32]. Along these lines, Hoenders and Slump gave in [11] a method for the determination of the total number of multiple roots of a single function. According to their method, if $f\colon (a, b) \subset \mathbb{R} \to \mathbb{R}$ is a k times continuously differentiable function, then the total number of zeros $\mathcal{N}_m^r$ of $f(x) = 0$ with multiplicity m or higher where $m \leq k$ can be obtained by the value of the topological degree calculated in a parallelepiped $\mathcal{P}$ by using in (2.7) as f_1 and f_2

$$\begin{aligned} f_1 &= f^2 + \sum_{l=1}^{m} \left(\frac{d^l f}{dx^l}\right)^2, \\ f_2 &= yf_1'. \end{aligned} \tag{2.11}$$

2.2. A modified bisection method. *Notation* 2.1. Throughout this paper the notation $\lceil\cdot\rceil$ refers to the smallest integer not less than the real number quoted; $\ell(I)$ indicates the length of the interval I.

It is well known that a solution of $f(x) = 0$ where the function f is continuous is guaranteed to exist in some interval $[a, b]$ if the following criterion is fulfilled:

$$f(a)\, f(b) \leq 0. \tag{2.12}$$

This criterion is known as Bolzano's existence criterion. Instead of Bolzano's criterion one may also use the value of the topological degree of f at origin relative to (a, b), which in this case can be defined as follows:

$$\deg[f, (a, b), 0] = \frac{1}{2}\left(\operatorname{sgn} f(b) - \operatorname{sgn} f(a)\right). \tag{2.13}$$

Now, if $\deg[f, (a, b), 0]$ is not zero, we know with certainty that there is at least one root in (a, b). Note that if $\deg[f, (a, b), 0]$ is not zero, then Bolzano's criterion is fulfilled. The value of $\deg[f, (a, b), 0]$ gives additional information concerning the behavior of the solutions of $f(x) = 0$ in (a, b) relative to the slopes of f [35]. For example, if $\deg[f, (a, b), 0] = 1$, which means that $f(b) > 0$ and $f(a) < 0$, then the number of solutions at points where $f(x)$ has a positive slope exceeds by one the number of solutions at points at which $f(x)$ has a negative slope.

Using the value of the topological degree (or, alternatively, Bolzano's criterion) we are able to calculate a solution of $f(x) = 0$ by bisecting the interval $I_0 = (a, b)$. So we subdivide I_0 into two intervals $(a, c], [c, b)$ where $c = (a + b)/2$ is the midpoint of (a, b) and we keep the subinterval for which the value of the topological degree is not zero relative to itself by checking the information on the boundaries. In this way we always keep at least one solution within a smaller interval. We can continue this procedure in order to approximate a solution until the endpoints of the final subinterval differ from each other by less than a fixed amount. This method is called the *bisection method* and can be generalized to higher dimensions [33, 34].

The main idea in order to locate all the solutions r_j, where $j = 1, \ldots, \mathcal{N}^r$ of $f(x) = 0$ in (a, b), is to subdivide the interval (a, b) and find $\mathcal{N}^r$ subintervals (a_j, b_j) for which the relation (2.12) is fulfilled. Now, for each one subinterval (a_j, b_j) we can apply any method to compute the root which is included in (a_j, b_j). Here we shall use the above-mentioned bisection method which has been modified to the following simplified version described in [33, 34]:

$$x_{i+1} = x_i + \operatorname{sgn} f(x_0) \operatorname{sgn} f(x_i)\, \ell(I_0)/2^{i+1}, \qquad x_0 = a, \qquad i = 0, 1, \ldots . \tag{2.14}$$

The sequence (2.14) converges to a root $r \in (a, b)$ if, for some x_i, $\operatorname{sgn} f(x_0) \operatorname{sgn} f(x_i) = -1$. Also, the number of iterations ν, which are required in obtaining an approximate root r^* such that $|r - r^*| \leq \varepsilon$ for some $\varepsilon \in (0, 1)$, is given by

$$\nu = \lceil \log_2(\ell(I_0)\, \varepsilon^{-1}) \rceil. \tag{2.15}$$

Instead of the iterative formula (2.14) we can also use

$$x_{i+1} = x_i - \operatorname{sgn} f(x_0) \operatorname{sgn} f(x_i)\, \ell(I_0)/2^{i+1}, \qquad x_0 = b, \qquad i = 0, 1, \ldots . \tag{2.16}$$

The bisection method always converges within the given interval (a, b) and it is a global convergence method. Moreover, it has a great advantage since it is optimal; i.e., it possesses asymptotically the best possible rate of convergence [27]. Also, using the relation (2.15) it is easy to have beforehand the number of iterations that are required for the attainment of an approximate root to a predetermined accuracy. Finally, it requires only the algebraic signs of the functions values to be computed, as is evident from (2.14) or (2.16); thus, it can be applied to problems with imprecise function values.

3. The algorithms. The algorithms are a sort of "guided" form of the bisection method. They use the outcome of relation (2.10) in order to isolate the roots or extrema by dividing the initial interval (a, b) into smaller intervals. Algorithm *find_roots*, described below in "pseudo Pascal," first determines the number of roots in the interval by applying (2.10) (step 8) and, if there are more than one, it divides the interval into m_n equal-size subintervals (steps 6 and 7) and proceeds recursively to each of the subintervals. We propose m_n to be equal to the number of roots though there are several issues to be considered here. We discuss this subject later on.

```
ALGORITHM find_roots($a$, $b$, $S$);
{comment: This algorithm locates and computes all the roots or all the extrema of $f(x) = 0$
  in $(a, b)$. It exploits (2.10) and (2.14). For (2.10) it requires $f$, $f'$, $f''$, and $\gamma$ while for (2.14)
  it requires $f$ and $\varepsilon$}
01. procedure roots($a, b, \mathcal{N}^r$); {comment: adds to set $S$ the $\mathcal{N}^r$ roots of the interval $(a, b)$}
      begin
02.     if $\mathcal{N}^r = 1$ then find the single root $r$ using the bisection (2.14), set $S \longleftarrow S \cup \{r\}$
        else
          begin
03.         $j \longleftarrow 1$; {comment: this counts the subintervals $I_j = (a_j, b_j)$}
04.         $k \longleftarrow 0$; {comment: this counts the computed roots}
05.         while $k < \mathcal{N}^r$ do
              begin
06.             $a_j \longleftarrow a + (j-1)\frac{b-a}{m_n}$; {comment: $m_n$ is the number of subintervals
                                                  in which we choose to divide $(a, b)$}
07.             $b_j \longleftarrow a + j\frac{b-a}{m_n}$;
08.             Find $\mathcal{N}_j^r$, the number of roots in $I_j$ using (2.10);
09.             if $\mathcal{N}_j^r > 0$ then roots($a_j, b_j, \mathcal{N}_j^r$);
```

```
            $k \longleftarrow k + \mathcal{N}_j^r$;
            $j \longleftarrow j + 1$
        **end** {while}
    **end**
  **end** {roots}
**begin** {find_roots}
  input $a, b$; {**comment**: $f(a)\, f(b)$ must be nonzero}
  $S \longleftarrow \emptyset$; {**comment**: $S$ is the set of roots in $(a, b)$}
  Find $\mathcal{N}_0^r$, the number of roots in $(a, b)$ using (2.10);
  roots($a, b, \mathcal{N}_0^r$);
  output $S$
**end.** {find_roots}
```

Remark 3.1. The case where only the extrema are required can be handled by the same algorithm by replacing the function f by its first derivative f'.

In the case where both roots and extrema are required, we can certainly apply the above method twice for f and f', respectively. But since the extrema lie between consecutive simple roots which are discovered by the first run of *find_roots* (for f), we now choose to divide initially at exactly the points of the roots. We next apply *find_roots* for f' for each subinterval between two consecutive roots. A high-level description of this algorithm follows.

```
ALGORITHM *roots_extrema*($a$, $b$, $S'$);
{**comment**: This algorithm locates and computes all the roots and all the extrema of $f(x)$
in $(a, b)$. It uses (2.10) and (2.14). For (2.10) it requires $f$, $f'$, $f''$, $f'''$, and $\gamma$ while for
(2.14) it requires $f$, $f'$, and $\varepsilon$}
**begin** {roots_extrema}
Find $\mathcal{N}^e$, the number of extrema in $(a, b)$ using (2.10);
Apply algorithm find_roots; Let $S = \{r_1, \dots, r_{\mathcal{N}^r}\}$ be its sorted output.
Construct the subintervals $I_j = (a_j, b_j) = (r_{j-1}, r_j)$, $j = 1, \dots, \mathcal{N}^r + 1$, $r_0 = a$,
  $r_{\mathcal{N}^r+1} = b$.
$E \longleftarrow \mathcal{N}^r - 1$; {**comment**: $E$ counts the number of verified extrema}
**if** $f'(a)\, f'(r_1) < 0$ **then** $E \longleftarrow E + 1$;
**if** $f'(r_{\mathcal{N}^r})\, f'(b) < 0$ **then** $E \longleftarrow E + 1$;
**if** $E = \mathcal{N}^e$ **then** apply bisection (2.14) (using $f'$) in all $I_j$. Let $S'$ the set of the extrema.
**else**
  **begin**
    $S' \longleftarrow \emptyset$;
    $j \longleftarrow 1$; {**comment**: this counts the subintervals $I_j = (a_j, b_j)$}
    $k \longleftarrow 0$; {**comment**: this counts the computed extrema}
    **while** $k < \mathcal{N}^e - E + j - 1$ **do**
      {**comment**: when $k + E - j + 1 = \mathcal{N}^e$, it is assured that
       only one extremum exists in each remaining interval}
      **begin**
        Apply algorithm *find_roots* in $I_j$ using $f'$; Let $S_j'$ be its output;
        Set $S' \longleftarrow S' \cup S_j'$; {**comment**: $S_j'$ will be the set of the extrema in $(a_j, b_j)$}
        $k \longleftarrow k + |S_j'|$;
        $j \longleftarrow j + 1$
      **end** {while}
    Apply bisection (2.14) (using $f'$) in all $I_l$ for $l = j + 1, \dots, \mathcal{N}^r$; set $S' \longleftarrow S' \cup S_l'$
  **end**
output $S'$
**end.** {roots_extrema}
```

Returning to algorithm *find_roots* and assuming that the number of roots in a subinterval is always available (say it is given by an "oracle"), it is clear that the computational complexity of the algorithm is determined by (1) the total number of calls to the oracle and (2) the iterations required by the $\mathcal{N}^r$ bisection calls which will compute the isolated roots.

4. The expected complexity of the algorithm. In this section we study the expected complexity of algorithm *find_roots*. First, we focus our attention on the number of times the number of roots has to be found in a given interval, i.e., the *average number of oracle calls* as this is the most demanding step of the localization phase. The rootfinding phase is dominated by the *average number of iterations* of the bisection method.

4.1. Preliminaries, definitions, and notations. The study presented here follows certain assumptions:

(i) The size of the problem is the total number of roots of $f(x)$ in (a, b).

(ii) We view each root of $f(x)$ as a random variable having a given distribution in (a, b). All roots are considered as pairwise independent variables having the same distribution. In this paper we assume that the distribution is *uniform*; i.e., it is equally likely for any point in the interval to be a root.

(iii) The study of the expected complexity of the algorithm is *over all possible inputs to the algorithm*, that is, over all possible sets of n points in the interval (a, b) and, consequently, independent of the given function.

Assumption (iii) implies only partial knowledge of the properties of the specific function (the distribution of its roots), which may vary from complete ignorance as to whether the roots lie (our case) to full knowledge of the roots if the properties of the function are known. Hence, the analysis is independent of the specific function which merely serves (through Picard's integral) as an "oracle" that reveals the number of roots in a specific interval. This models real life applications where some physical quantity is of interest whose zeros are statistically independent with known (or suspected) distribution. It is indeed intermediate cases of the form "the roots are expected around the center of the interval" that are really the most interesting since this type of knowledge may "tune up" appropriate algorithms that take into account the additional information and perform better. It is also worthwhile mentioning that the same analysis applies to any method that isolates the roots by repeated subdivisions of the intervals, independently of how the oracle is implemented (a good example is interval analysis techniques for which the main result of §4.2 is a lower bound). In this paper we completely ignore possible properties of $f(x)$ and solve the following combinatorial problem, leaving the more complete study for future research:

Find the expected number of oracle calls that algorithm find_roots *will require in order to isolate all n points which are randomly and independently chosen with uniform distribution.*

In other words, we want to find the expected value $\big($over all possible patterns of appearances of n roots in the interval $I_0 = (a, b)\big)$ of a function H which, given a specific pattern of roots in the interval, returns the number of oracle calls required to isolate each root in a subinterval (a formal definition follows). Since, however, this function is noncontinuous and we want to avoid integrating in subintervals, we adopt a simpler approach *discretizing* the interval $I_0 = (a, b)$ and summing instead of integrating.

We begin by giving a list of symbols and definitions that we shall need later.

Notation 4.1. Let $|S|$ denote the cardinality of the set S. Let $[\cdot]$ denote the integer part of the number quoted. Notice that $[x, y]$ refers to the closed interval with endpoints x and y.

DEFINITION 4.1. *We call resolution δ of the algorithm a small positive real such that if x is a root in (a, b), any point in the interval $\left[x - \frac{\delta}{2}, x + \frac{\delta}{2}\right]$ is considered to be the root x.*

This definition means that as far as the algorithm is concerned, any two roots less that δ apart are considered to be one and, as one, are reported by the oracle in (2.10). We next consider I_0 to be divided into a large number of consecutive subintervals of length δ which we call *elementary*. Any subdivision of I_0 into small intervals (as stated in steps 6 and 7 of the algorithm, as well as any subdivision thereof) is considered to take place at points that are *integer* multiples of δ. Going one step beyond, we consider (for simplicity reasons) I_0 to be rounded to the smallest integer multiple of δ, say μ, such that $a + \mu\delta \geq b$. Clearly, $\mu = \lceil \frac{b-a}{\delta} \rceil$. Consequently, if m roots are reported by the oracle at step 14 of the algorithm, steps 6 and 7 will divide I_0 with length $\ell(I_0)$ into m intervals $I_i, i = 1, \ldots, m$ such that

$$I_i = \left(a + \delta \left[(i-1)\frac{\ell(I_0)}{m\delta} \right], a + \delta \left[i \frac{\ell(I_0)}{m\delta} \right] \right], \quad i = 1, 2, \ldots, m.$$

In the above we adopt the convention that all subintervals be open from the left and closed from the right, thus avoiding a point belonging in more than one subinterval at each level of subdivision. Unifying these conventions, we consider the initial interval I_0 to be $I_0 = (a, a + \mu\delta]$.

The above discretization suggests a more convenient way of representing intervals, namely, as sets of the elementary consecutive subintervals which are included in them. More specifically, consider assigning to each elementary subinterval of I_0 an integer from 1 to μ in increasing order from left to right. This suggests representing I_0 by the set of consecutive integers $T_0 = \{l \in \mathbb{N} \colon 1 \leq l \leq \mu\}$. The representation of a subinterval I_i is

$$T_i = \left\{ l \in T_0 \colon \left[(i-1)\frac{\ell(I_0)}{m\delta} \right] + 1 \leq l \leq \left[i \frac{\ell(I_0)}{m\delta} \right] \right\}.$$

Analogous relations hold for any subinterval of I_0. For our study, T_0 is the sample space with each of its points a candidate of being a root. In the sequel we shall use the term "interval" both for a standard interval I and its corresponding T set when no confusion arises. Similarly, extending Notation 2.1, we denote by $\ell(T)$ the length of the corresponding standard interval I.

In the analysis that follows we shall denote by X a variable representing a set of integers (or, for our purposes, a set of roots). If $X \subset T_l$, X is any set of elementary subintervals, i.e., a set of integers in T_l. In the context described above, the probability of a set X of cardinality k, denoted by Prob$\{X\}$, is the probability of choosing the k elementary subintervals that correspond to the k roots. For example, if the distribution of the roots is uniform, then the probability of a specific X set is $\binom{\mu}{|X|}^{-1}$. Using the above, we are now ready to formally define the function H.

DEFINITION 4.2. *Let S be the set of all subsets of T_0. The function $H \colon X \in S \to \mathbb{N}$ maps X to the number of oracle calls that the algorithm will do in order to isolate the roots represented by the set X. Similarly, we define the H function relative to an interval T denoted by $H_T(X)$ and which gives the number of oracle calls that the algorithm will do in the specific interval T*; *that is, in the latter case we only count oracle calls required for roots inside T.*

Remark 4.1. If the interval T_0 is divided in m_n subintervals, T_i, $i = 1, \ldots, m_n$, the number of oracle calls in T_0 is clearly $H_{T_0} = 1 + \sum_{i=1}^{m_n} H_{T_i} - 1 = \sum_{i=1}^{m_n} H_{T_i}$. The "1" comes from the initial oracle call which returns the number of roots in T_0 and the "-1" from the fact that no oracle call is required to obtain the number of roots of the last interval since this can be obtained by subtracting from the total number the sum of the roots in the rest of the $m_n - 1$ subintervals. The same relation holds for the value of the H function of any interval. Note that here we assume that all $m_n - 1$ oracle calls will be required in the while loop of step 5.

4.2. Theoretical results. We now study the expected behavior of our algorithm beginning from the first stage, where the roots are isolated each in an interval by its own. Specifically, we study the *expected number of oracle calls* required for this task as a function of the number of roots n. Observe that the number of intervals into which we choose to divide can be either constant or a function of n. By n we denote the total number of roots in the given interval (n is the $\mathcal{N}_0^r$ of step 14 of the algorithm *find_roots*.) In any case, deciding the value of m_n is based only on n, and this is the reason why we denoted the number of subdivisions subscribed by n. As stated in Remark 4.1, the approach that follows assumes that all $m_n - 1$ calls are required in an interval which is divided into m_n subintervals. This is equivalent to replacing the while statement of step 5 by the statement **while** $j < m_n$ **do**. This is done for simplicity reasons and it is removed after the next theorem is proved.

THEOREM 4.1. *Suppose that the algorithm find_roots divides any interval T_0 with n roots into m_n subintervals $T_i, i = 1, \ldots, m_n$. Then the expected number of oracle calls required by the algorithm to isolate all n roots is given by the formula*

$$E_H(n) = {m_n}^{1-n} \sum_{k=0}^{n} \binom{n}{k} (m_n - 1)^{n-k} E_H(k) \tag{4.1}$$

or, equivalently,

$$E_H(n) = \sum_{k_1+\cdots+k_{m_n}=n} \frac{n!}{k_1!\cdots k_{m_n}!} {m_n}^{-n} \left(E_H(k_1) + \cdots + E_H(k_{m_n})\right), \tag{4.2}$$

where $E_H(0) = E_H(1) = 1$.

Proof. We shall show first (4.2). Clearly, $E_H(0) = E_H(1) = 1$ since, when no root or a single root is in T_0, then only one oracle call is required. For $n \geq 2$ let $X_i, i = 1, \ldots, m_n$ be the set of roots that lay in interval T_i. Clearly, $|X_1| + \cdots + |X_{m_n}| = n$. The distribution of the corresponding vector of the cardinalities of the X_i's $(|X_1|, \ldots, |X_{m_n}|)$ is polynomial; that is,

$$\text{Prob}\{|X_1| = k_1, \ldots, |X_{m_n}| = k_{m_n}\} = \frac{n!}{k_1!\cdots k_{m_n}!} {m_n}^{-n}. \tag{4.3}$$

This is immediate since Prob{some specific root lies in T_i} $= {m_n}^{-1}$ because of the uniform distribution of the roots in (a, b).

Let $\tilde{X} = X_1 \cup X_2 \cup \cdots \cup X_{m_n}$. By Remark 4.1 we have

$$H(\tilde{X}) = H_{T_1}(X_1) + H_{T_2}(X_2) + \cdots + H_{T_{m_n}}(X_{m_n}).$$

Taking the expectation [6] of both sides of the above we have

$$\begin{aligned} E\{H(\tilde{X})\} &= E\{H_{T_1}(X_1) + H_{T_2}(X_2) + \cdots + H_{T_{m_n}}(X_{m_n})\} \\ &= E\left\{E\{H_{T_1}(X_1) + H_{T_2}(X_2) + \cdots + H_{T_{m_n}}(X_{m_n}) \,\Big|\, |X_1| = k_1, \ldots, |X_{m_n}| = k_{m_n}\}\right\}, \end{aligned} \tag{4.4}$$

where the outer expectation is over all sets $\tilde{X} = X_1 \cup \cdots \cup X_{m_n}$ such that $|X_1| = k_1, \ldots, |X_{m_n}| = k_{m_n}$ with $k_1 + \cdots + k_{m_n} = n$. Now, the expectation of $H_{T_i}(X_i)$ where X_i runs over all possible choices of sets in T_i with $|X_i| = k_i$ depends only on the cardinality k_i of X_i. We may, therefore, simplify our notation and denote by $E_H(k_i)$ the expectation $E\{H_{T_i}(X_i)\}$. Note that we have also dropped the subscript T_i since the expectation does not depend on the specific interval either, as the distribution is uniform. Using linearity in the inner conditional expectation and the above observation we get

$$E_H(n) = E\left\{E_H(k_1) + \cdots + E_H(k_{m_n}) \,\Big|\, |X_1| = k_1, \ldots, |X_{m_n}| = k_{m_n}\right\}.$$

Using (4.3) we finally obtain

$$E_H(n) = \sum_{k_1+\cdots+k_{m_n}=n} \frac{n!}{k_1!\cdots k_{m_n}!} {m_n}^{-n} \left(E_H(k_1) + \cdots + E_H(k_{m_n})\right).$$

In order to show (4.1), observe that the above equation may be written as

$$\begin{aligned} E_H(n) &= \sum_{k=0}^{n} \binom{n}{k} {m_n}^{-k} \sum_{\substack{k_1+\cdots+k_{i-1}\\ +k_{i+1}+\cdots+k_{m_n}=n-k}} \binom{n-k}{k_1,\ldots,k_{i-1},k_{i+1},\ldots,k_m} {m_n}^{k-n} E_H(k) \\ &= \sum_{k=0}^{n} {m_n}^{-k} \binom{n}{k} \left(\frac{1}{m_n} + \cdots + \frac{1}{m_n}\right)^{n-k} E_H(k), \end{aligned}$$

where $m_n - 1$ fractions are added. Equation (4.1) now follows easily. □

Remark 4.2. Solving (4.1) with respect to $E_H(n)$ results in the following formula which can be used to obtain numerical values for $E_H(n)$:

$$E_H(n) = \left({m_n}^{n-1} - 1\right)^{-1} \sum_{k=0}^{n-1} \binom{n}{k} (m_n - 1)^{n-k} E_H(k). \tag{4.5}$$

An interesting question is raised as to whether we can remove recursion from (4.1). It turns out that recurrences of the above form are very difficult to handle. We were unable to do so in the general case, that is, when m is a function of n, or even in the case $m = n$, which is particularly interesting. However, when m is constant, we reduced (4.1) to a known recurrence solved in [18] by the use of *binomial transformations*. This is done by finding a recurrence for the sequence $E'_H(k) = E_H(k) - 1$, $k = 0, 1, \ldots$. By employing the techniques described in [18, p. 501] we can show that for $n \geq 2$

$$E'_H(n) = (m-1) \sum_{k=2}^{n} \binom{n}{k} \frac{(-1)^k (k-1) m^{k-1}}{m^{k-1} - 1}. \tag{4.6}$$

Observe that m is referred to without a subscript since the above hold when m is a constant.

We conclude the study of the first phase by examining the complexity of the algorithm *find_roots* as stated originally, that is, when step 5 includes the condition $k < \mathcal{N}^r$. Then we may be able to save several oracle calls if all the roots fall into the first few intervals and, consequently, it is unnecessary to call the oracle for the rest of the intervals. In order to calculate how much we overestimated the expected complexity in Theorem 4.1, observe that in (4.4) the expectation is over every possible $\tilde{X}$ with cardinality n. But if $\tilde{X}$ totally falls in intervals T_1 to T_i, algorithm *find_roots* will not require oracle calls for the last $m_n - i$ intervals. So the expectation was overestimated by $(m_n - i) \times \text{Prob}\{\tilde{X}$ totally falls in intervals T_1 to $T_i\}$. For example, when $\tilde{X}$ is totally included in the first interval, then a term $(\frac{1}{m_n})^n (m_n - 1)$ must be subtracted. Similarly, an additional $(m_n - 2)\left((\frac{2}{m_n})^n - (\frac{1}{m_n})^n\right)$ must be subtracted when $\tilde{X}$ is totally included in the first two intervals. The subtracted $(\frac{1}{m_n})^n$ is the probability of $\tilde{X}$ to fall totally in the first interval and was already counted in the term $(\frac{1}{m_n})^n (m_n - 1)$. Hence, we must decrease (4.1) by

$$\sum_{i=1}^{m_n} \left[\left(\frac{i}{m_n}\right)^n - \left(\frac{i-1}{m_n}\right)^n \right] (m_n - i) = \sum_{i=1}^{m_n - 1} \left(\frac{i}{m_n}\right)^n .$$

We, therefore, have the following theorem.

THEOREM 4.2. *The expected number of oracle calls of algorithm find_roots is given by*

$$E_H(n) = m_n^{1-n} \sum_{k=0}^{n} \binom{n}{k} (m_n - 1)^{n-k} E_H(k) - \sum_{i=1}^{m_n - 1} \left(\frac{i}{m_n} \right)^n . \tag{4.7}$$

Proof. The proof follows immediately from Theorem 4.1 and the above discussion. □

We next proceed to the study of the second phase of the algorithm when all the roots have been isolated, each in its own interval. This second phase consists of a number of bisections that are applied in those intervals. Our task, therefore, requires the definition of a complexity function that will capture the actual work of the second phase, like the H function defined above. A promising approach might be to use the *average length* of the n intervals that are searched by bisection. (Note here that we are talking about the average length of the intervals of a *specific* run of the algorithm and not over every possible run. Should we accept this approach, we must next calculate the *expected average length* of the intervals.) But there is a subtle point here. The actual average case work of the bisection is logarithmically related to the length of the interval in which it is applied (see §2). This means that if we want to estimate the work of the second phase, the average (the arithmetic mean, to be precise) length of the intervals is not the best measure. In fact, it is not difficult to see that the *geometric mean* is a quantity proportional to the total number of iterations required. Seeking for an appropriate complexity measure we give the following definition.

DEFINITION 4.3. *Let $\tilde{X}$ be the set of roots and T any subinterval of T_0. We denote by $SP[T;\tilde{X}]$ the set of subsets of T into which the algorithm will divide T and that include one element of $\tilde{X}$.*

In other words, $\mathrm{SP}[T;\tilde{X}]$ is the set of subintervals of T produced by the algorithm each containing a single root and to which bisection must be applied. Observe now that the total number of iterations IT, involved in phase two, is given by

$$\mathrm{IT} = \sum_{T \in \mathrm{SP}[T_0;\tilde{X}]} \log_2 \left(\ell(T)\, \varepsilon^{-1} \right).$$

The above formula has the disadvantage that it involves the absolute length of the intervals that remains after phase one, which makes it inappropriate for a recursive relation. By a slight modification we get the following relation which, contrary to the above, involves the relative (with respect to the initial interval T_0) lengths of the intervals:

$$\mathrm{IT} = \sum_{T \in \mathrm{SP}[T_0;\tilde{X}]} \log_2 \frac{|T|}{|T_0|} + n \log_2 \left(\ell(T_0)\, \varepsilon^{-1} \right). \tag{4.8}$$

Observe that if we manage to estimate the average value of the sum, then finding the expected number of iterations is a trivial task. Therefore, we focus our attention to this sum which we call the "characteristic complexity function of phase two" and define it as follows.

DEFINITION 4.4. *Let $\tilde{X}$ be a set of roots in a specific instance of the problem. The "characteristic complexity function" of phase two is defined by*

$$B(\tilde{X}) = \sum_{T \in \mathrm{SP}[T_0;\tilde{X}]} \log_2 \frac{|T|}{|T_0|}.$$

Similarly, we define the function B relative to an interval T' and denote it by $B_{T'}$ to be

$$B_{T'}(\tilde{X}) = \sum_{T \in \mathrm{SP}[T';\tilde{X}]} \log_2 \frac{|T|}{|T'|}.$$

Our next task will be to calculate the expected value of B.

THEOREM 4.3. *Suppose that the algorithm divides an interval T_0 with n roots into m_n subintervals. Then the expected value of the characteristic complexity function is given by the formula*

$$E_B(n) = {m_n}^{1-n} \sum_{k=0}^{n} \binom{n}{k} (m_n - 1)^{n-k} E_B(k) - n\log_2 m_n \tag{4.9}$$

or, equivalently,

$$E_B(n) = \sum_{k_1+\cdots+k_{m_n}=n} \frac{n!}{k_1!\cdots k_{m_n}!} {m_n}^{-n} \left(E_B(k_1) + \cdots + E_B(k_{m_n})\right) - n\log_2 m_n, \tag{4.10}$$

where $E_B(0) = E_B(1) = 0$.

Proof. First observe that $E_B(0) = 0$ since $\mathrm{SP}[T;\emptyset] = \emptyset$. Also, $E_B(1) = 0$ since when only one root is in an interval, the whole interval must be searched by the bisection. For any $n \geq 2$ let, as in Theorem 4.1, $X_i, i = 1, \ldots, m_n$ be the set of roots in intervals $T_i, i = 1, \ldots, m_n$, respectively. Also, let $\tilde{X}$ be the total set of roots in T_0. Then

$$\begin{aligned} B(\tilde{X}) &= \sum_{T\in \mathrm{SP}[T_0;\tilde{X}]} \log_2 \frac{|T|}{|T_0|} = \sum_{T\in \mathrm{SP}[T_1;X_1]} \log_2 \frac{|T|}{|T_0|} + \cdots + \sum_{T\in \mathrm{SP}[T_{m_n};X_{m_n}]} \log_2 \frac{|T|}{|T_0|} \\ &= \sum_{T\in \mathrm{SP}[T_1;X_1]} \log_2 \frac{|T|}{|T_1|} - k_1\log_2 m_n + \cdots + \sum_{T\in \mathrm{SP}[T_{m_n};X_{m_n}]} \log_2 \frac{|T|}{|T_{m_n}|} - k_{m_n}\log_2 m_n, \end{aligned}$$

where $k_1, \ldots, k_{m_n}$ are the numbers of roots in intervals $T_1, \ldots, T_{m_n}$, respectively. Consequently, under the above assumptions

$$B(\tilde{X}) = B_{T_1}(X_1) + B_{T_2}(X_2) + \cdots + B_{T_{m_n}}(X_{m_n}) - n\log_2 m_n. \tag{4.11}$$

The key observation in (4.11) is, once again, that the expectations of the B functions are independent of the intervals in which they are applied and depend only on the cardinalities of the corresponding X set. Moreover, the subtracted term $n\log_2 m_n$ is independent of the X sets. Hence, by taking the expectations of both sides of (4.11) we may apply the same line of thought as in Theorem 4.1 and obtain (4.10). Proving (4.9) is then completely analogous to proving relation (4.1). □

Remark 4.3. Solving (4.9) with respect to $E_B(n)$ we get the following formula which may be used to obtain numerical values for $E_B(n)$:

$$E_B(n) = \frac{1}{{m_n}^{n-1} - 1} \sum_{k=0}^{n-1} \binom{n}{k} (m_n - 1)^{n-k} E_B(k) - n{m_n}^{n-1}\log_2 m_n. \tag{4.12}$$

COROLLARY 4.4. *The expected number of iterations required by the algorithm in order to compute all n roots is given by the formula*

$$\mathrm{IT} = E_B(n) + n\log_2\left(\ell(T_0)\,\varepsilon^{-1}\right). \tag{4.13}$$

Proof. The proof follows from (4.8) and Theorem 4.3. $E_B(n)$ must be computed from (4.12). □

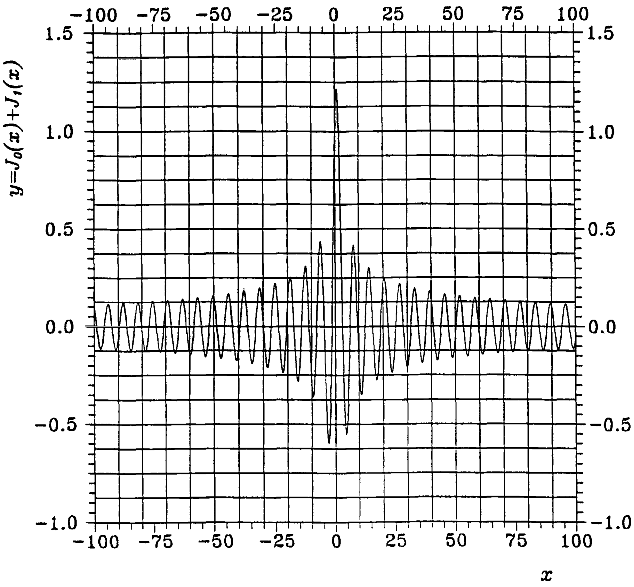


FIG. 1. *The function* (5.1) *for* $c = 0$, $a = -100$, *and* $b = 100$.

5. An example and numerical applications. We illustrate our method and analysis on a parametric problem based on a well-known function also studied in [10]:

$$f(x) = J_0(x) + J_1(x) + c, \tag{5.1}$$

where J_0 and J_1 indicate the zero-order and first-order Bessel functions [26], respectively, and c a constant, where

$$J_n(x) = \sum_{k=0}^{\infty}(-1)^k \frac{(x/2)^{2k+n}}{k!(n+k)!}, \quad n = 0, 1, \dots . \tag{5.2}$$

Function (5.1) was selected as this and related functions attract the attention of several researchers. Moreover, it possesses some nice properties which make it especially suitable for our purposes. It has a large number of roots which means that arbitrarily large rootfinding problems may be generated, restricted only by the boundaries of the interval. Also, the roots of the function are within a small neighborhood almost equally spaced, which, in turn, results in a performance of the algorithm close to that predicted by the analytical estimations. Finally, as it is obvious from Figure 1, by varying the constant c we can, in effect, raise or lower the function relative to the x–axis and, consequently, "move" the roots toward one end of the interval and study the results in the performance of the algorithm. We applied the algorithm *find_roots* for various intervals in order to compute the corresponding roots. The behavior of the algorithm for the extrema is analogous.

Using the relations

$$\begin{aligned} J_0'(x) &= -J_1(x), \\ J_1'(x) &= J_0(x) - \frac{1}{x}J_1(x), \end{aligned} \tag{5.3}$$

we are able to find the derivatives used in (2.10) as functions of $J_0(x)$ and $J_1(x)$ which, in turn, may be computed using, for example, the routines of [26].

TABLE 1
Computer runs and behavior of the algorithm find_roots for various instances of the problem.

Function	a	b	$\mathcal{N}^r$	OC	IT	EOC	EIT
$J_0(x) + J_1(x)$	-1000	1000	636	636	26712	1027.9	25438.4
$J_0(x) + J_1(x)$	-100	100	63	63	2646	101.1	2521.5
$J_0(x) + J_1(x)$	0	100	31	31	1302	49.3	1241.9
$J_0(x) + J_1(x) - 0.125$	-100	100	50	67	2194	80.0	2018.0
$J_0(x) + J_1(x) - 0.125$	0	100	25	32	1205	39.6	1009.4
$J_0(x) + J_1(x) - 0.125$	0	200	25	40	1611	39.6	1034.4
$J_0(x) + J_1(x) - 0.125$	0	300	25	43	1760	39.6	1049.1
$J_0(x) + J_1(x) - 0.15$	-100	100	34	50	1522	54.2	1392.4
$J_0(x) + J_1(x) - 0.15$	0	100	17	25	933	26.7	696.2
$J_0(x) + J_1(x) - 0.15$	0	200	17	29	1202	26.7	713.2
$J_0(x) + J_1(x) - 0.15$	0	300	17	31	1302	26.7	723.1

Using bisection for this kind of problem has the advantage that only signs need to be computed, which can be achieved by considering relatively few terms of (5.2).

The results for locating and computing all the zeros are summarized in Table 1. The columns labeled a and b indicate the endpoints of the searched interval; the columns labeled $\mathcal{N}^r$, OC, and IT show the number of computed roots within (a, b), the number of oracle calls, and the number of iterations required by the algorithm *find_roots*, respectively. The two final columns EOC and EIT indicate the values of (4.7) and (4.13) for the same parameters.

As it is evident from Table 1, the predictions from (4.7) and (4.13) are closer to the actual performance in the cases where the pattern of the roots has a certain degree of "asymmetry" that captures the possible variations from uniformity.

6. Discussion and open problems. Let us now focus our attention on the results of §4 and discuss the expected behavior of the algorithm as described by (4.7) and (4.12). The first equation describes the behavior of phase one of the algorithm as a function of n, the number of roots. In Figure 2 we plot (4.7) for various values of m_n, the number of subintervals in which we divide. It is clear that the expected number of calls decreases when we decrease the number of subintervals with a minimum reached for $m_n = 2$. An interesting choice seems to be $m_n = n$ since the curve for this case remains close to the curve of $m_n = 2$.

The reverse holds for the number of iterations of phase two which is plotted in Figure 3. There we plot (4.12) since this is actually the expected number of iterations "saved" by this method (this is represented by the negative sign of the E_B values) compared with n bisections on the interval T_0. When the actual number of iterations is desired, then the term $n \log_2 \left(\ell(T_0)\, \varepsilon^{-1}\right)$ must be added. It is clear, therefore, that what we save must be compared with this last term. Figure 3 shows that the saved iterations grow roughly like $-10n$ (for $m_n = 2$). The added term depends on the quantity $\ell(T_0)\, \varepsilon^{-1}$. Assuming a value of 10^{12} reasonable for this term, i.e., 12 significant digits (this corresponds to 12 decimal digits for the normalized case $\ell(T_0) = 1$), we see that the added term grows like $40n$. Hence, we save about 25% of the iterations. Less demanding accuracy, for example, for $\ell(T_0)\, \varepsilon^{-1} = 10^6$, results in a savings of about 50%. (At this point one may wonder whether the negative part of (4.13) exceeds, in absolute value, the positive part for suitable $\ell(T_0)$ and ε. This is not the case since the analysis above presumes that each subinterval is greater than ε, the desired accuracy. Hence, (4.13) holds for sufficiently small ε.)

It is clear, therefore, that the total complexity of the algorithm is a combination of the complexities of phases one and two. A central issue is the choice of m_n:. phase one requires small m_n, if possible two, but phase two requires large m_n. A reasonable choice (which is also the one proposed in the algorithms) seems to be $m_n = n$.

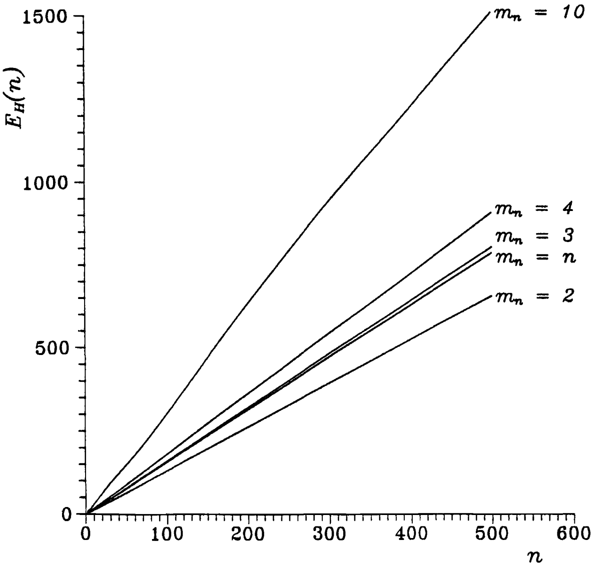


FIG. 2. *Expected number of oracle calls by virtue of* (4.7) *for various values of* m_n.

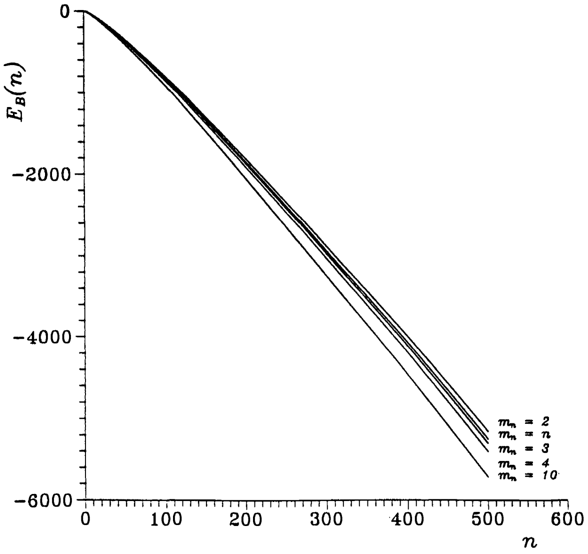


FIG. 3. *Expected number of iterations saved by virtue of* (4.12) *for various values of* m_n.

Every “real life” estimation, however, heavily depends on the way the “oracle” is implemented. The central part of (2.10) is clearly the integral. If the integral can be implemented in a nontime-consuming way (for example, when the analytical form can be found or an efficient numerical method can be applied), then phase one will be less demanding and we must concentrate on phase two by increasing the m_n value. If, however, this is not the case, then probably small values of m_n would be preferred. In any case, the integral must be computed with an error of, at most, 0.5 since it is known that (2.10) returns an integer. This suggests that relatively few function evaluations will be necessary. And, most importantly, the implementation of the oracles for the subintervals can take into account the function evaluations that have already been computed for the oracle of the divided interval. This last observation suggests that numerical values should be stored and reused for the oracles of higher levels of subdivision. Such an approach could dramatically reduce the total number of function evaluations for the oracles implementation. The full exploitation of this idea and the trade-off between phases one and two will be presented in a future publication.

Several other open problems remain to be examined in different directions. A first direction is to apply the method for more general distributions of the roots. This will, first of all, broaden the class of problems for which the analytical estimations apply. Next, one may apply the same techniques and derive complexity bounds for other methods on the same problem along the lines of Theorems 4.1 and 4.3. But we also feel that there are several things to be examined here in the algorithm itself. For example, if we expect the roots to be concentrated around the middle of the interval, then clearly more refined subdivisions must take place there, while close to the boundaries the subdivision can be sparser. That is, we believe that the algorithm itself can be guided if the distribution of the roots is known, in order to improve its performance. Results on this direction will be reported in [12].

Furthermore, preliminary investigations suggest that the algorithms can be generalized to higher dimensions to locate and compute with certainty all the simple roots of equations $F(x) = (0, 0, \dots, 0)$ where $F = (f_1, f_2, \dots, f_n)\colon (a, b)^n \subset \mathbb{R}^n \to \mathbb{R}^n$, as well as all the extrema of functions $f\colon (a, b)^n \subset \mathbb{R}^n \to \mathbb{R}$.

Finally, we would like to point out the possibilities of efficient parallelization of the algorithm, thus exploiting the tremendous capabilities of modern parallel computers. This opens a totally new subject where all the discussed problems should be reexamined in the light of massive parallelism.

Acknowledgments. The authors wish to thank Stavros Kourouklis for useful discussions that led to an alternative proof of Theorem 4.1 and Dimitris Sotiropoulos for programming assistance.

REFERENCES

[1] G. Alefeld and J. Herzberger, *Introduction to Interval Computations*, Academic Press, New York, 1983.

[2] G. Alefeld, F. A. Potra, and Yixun Shi, *On enclosing simple roots of nonlinear equations*, Math. Comp., 61 (1993), pp. 733–744.

[3] P. Alexandroff and H. Hopf, *Topologie*, Springer-Verlag, Berlin, 1935; reprinted Chelsea, New York, 1965.

[4] T. Boult and K. Sikorski, *An optimal complexity algorithm for computing the topological degree in two dimensions*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 686–698.

[5] A. Davidoglou, *Sur le nombre des racines communes à plusieurs équations*, Compt. Rend. hebd., 133 (1901), pp. 784–786 and pp. 860–863.

[6] W. Feller, *An Introduction to Probability Theory and Its Applications*, Vol. I, John Wiley, New York, 1968.

[7] E. R. Hansen, *A globally convergent interval method for computing and bounding real roots*, BIT, 18 (1987), pp. 415–424.

[8] ———, *Global Optimization Using Interval Analysis*, Marcel Dekker, New York, 1992.

[9] E. R. Hansen and G. W. Walster, *Nonlinear equations and optimization*, Comput. Math. Appl., 25 (1993), pp. 125–145.

[10] B. J. Hoenders and C. H. Slump, *On the calculation of the exact number of zeros of a set of equations*, Computing, 30 (1983), pp. 137–147.

[11] ———, *On the determination of the number and multiplicity of zeros of a function*, Computing, 47 (1992), pp. 323–336.

[12] D. J. Kavvadias and M. N. Vrahatis, *Locating and computing all roots offunctions with arbitrarily distributed roots*, manuscript.

[13] R. B. Kearfott, *Computing the Degree of Maps and a Generalized Method of Bisection*, Ph.D. thesis, Department of Mathematics, University of Utah, Salt Lake City, UT, 1977.

[14] ———, *An efficient degree-computation method for a generalized method of bisection*, Numer. Math., 32 (1979), pp. 109–127.

[15] ———, *Some tests of generalized bisection*, ACM Trans. Math. Software, 13 (1987), pp. 197–220.

[16] R. B. KEARFOTT AND M. NOVOA III, *INTBIS: A portable interval Newton/bisection package*, ACM Trans. Math. Software, 16 (1990), pp. 152–157.

[17] R. B. KEARFOTT, *Interval arithmetic techniques in the computational solution of nonlinear systems of equations: Introduction, examples, and comparisons*, Lectures in Appl. Math., 26 (1990), pp. 337–357.

[18] D. E. KNUTH, *The Art of Computer Programming, Vol.* 3: *Sorting and Searching*, Addison–Wesley, Reading, MA, 1973.

[19] L. KRONECKER, *Werke*, Vol. 1, Teubner, Leipzig, 1895.

[20] D. LE, *An efficient derivative-free method for solving nonlinear equations*, ACM Trans. Math. Software, 11 (1985), pp. 250–262.

[21] ———, *Three new rapidly convergent algorithms for finding a zero of a function*, SIAM J. Sci. Statist. Comput., 16 (1985), pp. 193–208.

[22] T. O'NEIL AND J. THOMAS, *The calculation of the topological degree by quadrature*, SIAM J. Numer. Anal., 12 (1975), pp. 673–680.

[23] J. M. ORTEGA AND W. C. RHEINBOLT, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.

[24] E. PICARD, *Sur le nombre des racines communes à plusieurs équations simultanées*, J. Math. Pures Appl. (4^e série), 8 (1892), pp. 5–24.

[25] ———, *Traité d'analyse*, 3rd ed., Gauthier–Villars, Paris, 1922.

[26] W. H. PRESS, B. P. FLANNERY, S. A. TEUKOLSKY, AND W. T. VETTERLING, *Numerical Recipes, The Art of Scientific Computing*, Cambridge University Press, Cambridge, 1992.

[27] K. SIKORSKI, *Bisection is optimal*, Numer. Math., 40 (1982), pp. 111–117.

[28] C. H. SLUMP AND B. J. HOENDERS, *The determination of the location of the global maximum of a function in the presence of several local extrema*, IEEE Trans. Inform. Theory, 31 (1985), pp. 490–497.

[29] F. STENGER, *Computing the topological degree of a mapping in* $\mathbb{R}^n$, Numer. Math., 25 (1975), pp. 23–38.

[30] M. STYNES, *An Algorithm for the Numerical Calculation of the Degree of a Mapping*, Ph.D. thesis, Department of Mathematics, Oregon State University, Corvallis, OR, 1977.

[31] ———, *An algorithm for numerical calculation of topological degree*, Appl. Anal., 9 (1979), pp. 63–77.

[32] G. TZITZÉICA, *Sur le nombre des racines communes à plusieurs équations*, Compt. Rend. hebd., 133 (1901), pp. 918–920.

[33] M. N. VRAHATIS, *Solving systems of nonlinear equations using the nonzero value of the topological degree*, ACM Trans. Math. Software, 14 (1988), pp. 312–329.

[34] ———, *CHABIS: A mathematical software package for locating and evaluating roots of systems of nonlinear equations*, ACM Trans. Math. Software, 14 (1988), pp. 330–336.

[35] ———, *A short proof and a generalization of Miranda's existence theorem*, Proc. Amer. Math. Soc., 107 (1989), pp. 701–703.

SIAM J. SCI. COMPUT.
Vol. 17, No. 5, pp. 1249–1268, September 1996

014

ON WEAK RESIDUAL ERROR ESTIMATION*

JINN-LIANG LIU†

Abstract. A general framework for weak residual error estimators applying to various types of boundary value problems in connection with finite element and finite volume approximations is developed. Basic ideas commonly shared by various applications in error estimation and adaptive computation are presented and illustrated. Some numerical results are given to show the effectiveness and efficiency of the estimators.

Key words. adaptivity, a posteriori error estimates, finite element, finite volume, variational problems

AMS subject classifications. 65N30, 65N50, 35J20, 35J60

1. Introduction.

Adaptivity is a trend in contemporary computational science. The remarkable advances in adaptive methodology for finite element applications since the pioneer work by Babuška and Rheinboldt [7] have had a profound impact on practical, large-scale computations.

There are four types of adaptivity which can be identified by the letters r (relocating nodes), h (mesh size h refinement), p (spectral order p increment), and hp (both h refinement and p increment). All adaptive methods require more or less a posteriori information about the computed solution for optimizing overall computational efforts in the sense that the methods deliver a given level of accuracy with a minimum of degrees of freedom. In essence, the *a posteriori error estimation* can be regarded as the heart of the adaptive mechanism. In the development of a posteriori error estimators, three main approaches may be distinguished [22], [29], namely, those based on residual, postprocessing, or interpolation techniques. Our estimators here follow the first approach.

In the first approach it has become practically standard to specify the interior residuals in terms of the governing differential equation and to measure the boundary residuals by the jumps in the normal derivatives on the interfaces between elements; see [3], [4], [6], [7], [8], [11], [19]. The various error estimators then differ essentially in the way the jumps are handled. In contrast, we consider here error equations (or inequalities) in which the right side is a residual of the computed solution in weak form. It appears to be natural to call the resulting error estimates *weak residual* estimators. A special case of the estimators seems to be first proposed by Adjerid and Flaherty in [1]. The weak residual error estimators tend to be more widely applicable since, in many applications, the governing equation may not be available in differential form. In recent years, there has been growing evidence, in theory as well as in application, showing the promise of the use of weak residual estimators; see [1], [2], [9], [21], [25].

This paper attempts to give an overall view of the weak residual error estimation in connection with the adaptive process, different numerical methods, and various types of boundary value problems. The numerical schemes of particular interest belong to two families of widely used methods—the finite element method (FEM) and the finite volume method (FVM). All estimators presented here can be extracted to two basic components which are *weak residuals* and *complementary finite element (FE) spaces*. Two types of complementary spaces can be classified. One is the conforming type which together with the original FE space preserves the continuity across adjacent FEs. This in turn corresponds to an elementwise error estimation using error residuals only interior to elements. The estimators used in [1], [2], [9],

*Received by the editors May 24, 1993; accepted for publication (in revised form) April 24, 1995. This work was supported in part by NSC grant 82-0208-M-009-060, Taiwan, ROC.

†Department of Applied Mathematics, National Chiao Tung University, Hsinchu, Taiwan (jinnliu@math.nctu.edu.tw).

[25] are of this type. The other is the nonconforming type which instead uses both interior and boundary residuals for each element. It is shown in [21] that the nonconforming type is independent of the FE order used for the computed solution, whereas the conforming type considers more closely the effect of the order of the FEs [1], [8]. While this study is not comprehensive, we do illustrate the two components in error estimation and the adaptive process by four model problems; namely, we consider elliptic boundary value problems, parametrized nonlinear equations, symmetric hyperbolic equations, and variational inequalities.

In §2, the error estimators are derived in a general framework by means of abstract variational formulations. Definitions, notation, and basic ideas are then introduced along the course of the derivation. Although the estimators are not restricted to any particular type of adaptivity, we also discuss briefly a standard h-refinement strategy in two space dimensions. In §3, the four model problems and the two numerical methods are specifically used to demonstrate how they can be cast into the general framework. Finally, in §4, numerical results are given with respect to each subsection in §3 to show the effectiveness and efficiency of the estimators.

2. Background and basic ideas. The aim of this paper is to offer, to the extent possible, a global view of the use of weak residual error estimators. It is instructive to summarize some fundamental features which constitute various adaptive methods for various model problems considered herein.

2.1. Abstract variational problems. Let Ω be a bounded region in the plane with a Lipschitz boundary $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$ and $H^k(\Omega)$ and $H^r(\Gamma)$, $\Gamma \subset \partial\Omega$, be the usual Sobolev spaces equipped with the norms $\|\cdot\|_k$ and $|\cdot|_{r,\Gamma}$, respectively. Let $H(\Omega) \subset L^2(\Omega)$ be a Sobolev space equipped with the norm $\|\cdot\|$. For simplicity, we assume that all functions in $H(\Omega)$ satisfy a homogeneous Dirichlet boundary condition on $\partial\Omega_D$, if any. Let K be a closed convex subset of $H(\Omega)$. Let $F : H(\Omega) \to R$ be a continuous linear form. Let $B(\cdot,\cdot) : H(\Omega) \times H(\Omega) \to R$ be a bilinear form such that there exist two positive constants β, δ and a nonnegative constant α for which

$$B(u,v) \le \beta\|u\|\,\|v\|, \quad u, v \in H(\Omega), \tag{2.1}$$

$$B(u,u) \ge \delta\|u\|^2 - \alpha\|u\|_0^2, \quad u \in H(\Omega). \tag{2.2}$$

We shall consider a class of very general problems in an abstract variational setting: Find $u \in K$ such that

$$B(u,v) - F(v) \ge B(u,u) - F(u) \quad \forall v \in K. \tag{2.3}$$

Depending on the definition of the closed convex set K and the properties of the bilinear form B, the abstract variational problem (2.3) can give various equivalent formulations of problems which will be exemplified in §3. Discussions of well-posedness of the problem (2.3), in some selective settings, can be found, e.g., in [13], [18].

2.2. FE spaces. To discretize (2.3), we introduce S a finite-dimensional subspace of $H(\Omega)$ characterized by a mesh size h and associated with a regular, but not necessarily quasi-uniform, triangulation $\mathcal{T}$ on Ω. To approximate K, we construct a closed convex subset K_s of S. The approximate problem of (2.3) is then to find $u_s \in K_s$ such that

$$B(u_s, v_s) - F(v_s) \ge B(u_s, u_s) - F(u_s) \quad \forall v_s \in K_s. \tag{2.4}$$

Our objective is to present various error formulations on which various a posteriori error estimators assessing the exact error between the solutions u and u_s of (2.3) and (2.4), respectively, are based. All estimators can be extracted to two basic components—weak residuals and complementary FE spaces. We first discuss the complementary spaces.

For the sake of efficiency, the estimators will be based on local computations. We introduce some local function spaces that we will require. Associated with $\mathcal{T}$, let $\bar{S}$ be another larger FE subspace of $H(\Omega)$, i.e., $S \subset \bar{S}$. Let $H(\tau)$ denote the restriction of $H(\Omega)$ to $\tau \in \mathcal{T}$ and $H_{\mathcal{T}}(\Omega) = \prod_{\tau \in \mathcal{T}} H(\tau)$ denote the space of piecewise $H(\Omega)$ functions. For instance, if $H(\Omega) \equiv H^1(\Omega)$, then $H_{\mathcal{T}}(\Omega)$ will be the space of piecewise H^1 functions. For v, w in $H_{\mathcal{T}}(\Omega)$, we define the broken L^2 inner product and norm by $(v, w) = \sum_{\tau \in \mathcal{T}} (v, w)_\tau$, $\|v\|_0^2 = (v, v)$, analogously, the broken $H(\Omega)$ norm, $\|v\|^2 = \sum_{\tau \in \mathcal{T}} \|v\|_{H(\tau)}^2$. Note that $H(\Omega) \subset H_{\mathcal{T}}(\Omega) \subset L^2(\Omega)$ and the above definitions reduce to the usual ones whenever v, w are in $H(\Omega)$. Let $\bar{S}$ be split into two subspaces $\bar{S} = S \oplus S^c$, $S \cap S^c = \{0\}$. Let S_τ^c denote the restriction of S^c to $\tau \in \mathcal{T}$ and let $S_{\mathcal{T}}^c = \prod_{\tau \in \mathcal{T}} S_\tau^c$. We shall consider in particular the splitting

$$\bar{S}_{\mathcal{T}} = S \oplus S_{\mathcal{T}}^c, \quad S \cap S_{\mathcal{T}}^c = \{0\}, \quad S_{\mathcal{T}}^c \neq \emptyset. \tag{2.5}$$

The spaces $S_{\mathcal{T}}^c$ and $\bar{S}_{\mathcal{T}}$ are spaces of piecewise polynomials locally defined in each element τ in $\mathcal{T}$. Note the inclusions $\bar{S} \subset \bar{S}_{\mathcal{T}} \subset H_{\mathcal{T}}(\Omega)$ and $S^c \subset S_{\mathcal{T}}^c$. The error estimators given in §3 will be calculated in the complementary space $S_{\mathcal{T}}^c$. We assume that the bilinear form B can define an inner product $B(\cdot, \cdot)$ on $\bar{S}_{\mathcal{T}}$ and with it the energy norm $|||w|||^2 = B(w, w)$. All error estimators below are measured in this norm, although, in theory, they are not strictly restricted to this norm.

2.3. Abstract variational error formulation. Let $e = u - u_s$ denote the exact error of the approximate solution u_s. We now derive a general formulation for the exact error in terms of the approximate solution. Substituting $u = e + u_s$ into problem (2.3), we have

$$\begin{aligned} &B(e, v) - F(v) + B(u_s, v) \\ &\geq B(e, e) + B(e, u_s) + B(u_s, e) + B(u_s, u_s) - F(e) - F(u_s) \quad \forall v \in K. \end{aligned}$$

Note that the bilinear form B can be nonsymmetric. By rearranging terms in the above inequality, we can rewrite this as

$$\begin{aligned} &B(e, v - u_s) - [F(v - u_s) - B(u_s, v - u_s)] \\ &\geq B(e, e) - [F(e) - B(u_s, e)] \quad \forall v \in K. \end{aligned}$$

The left-hand side of the inequality clearly suggests that we can define the following new closed convex subset by translating the original convex set K with respect to the computed solution u_s:

$$K' = K - u_s \subset H(\Omega).$$

Moreover, by virtue of the boundedness of the bilinear form B and the linear functional F on $H(\Omega)$, the Riesz representation theorem shows that there exists a unique linear functional G on $H(\Omega)$ defined by

$$G(w) = F(w) - B(u_s, w) \ \forall w \in H(\Omega). \tag{2.6}$$

We call G a *weak residual* in contrast to the usual formulation in which the residual is in terms of the governing differential equation used by many authors [8], [11], [19].

The error estimation is then based on the following new variational (error) problem: Determine $e \in K'$ such that

$$B(e, w) - G(w) \geq B(e, e) - G(e) \quad \forall w \in K'. \tag{2.7}$$

The reason we use the weak residual form G instead of the governing equation is twofold. First, since u_s is itself an approximate solution, its second derivative, for second-order partial

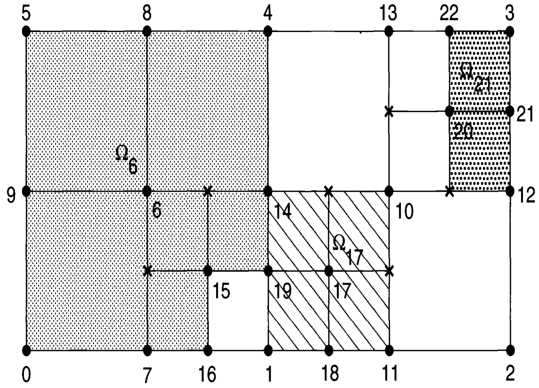


FIG. 2.1. *1-irregular mesh refinement.*

differential equations (PDEs), in the sense of distribution will further incur error. Second, for many applications, the governing differential equation may not be available, i.e., only the integral form is available. For such cases, the formulation of error problems like (2.7) is certainly more general.

2.4. A mesh refinement strategy. There are many refinement strategies proposed in the literature. The error estimators presented here are not restricted to any particular adaptive refinement. In fact, it has been shown in [21] that there is no order restriction for the FE spaces used in the approximate solution or in the error estimation; that is, the error estimators can be used in connection with any one of the h-, p-, or hp-versions of the FEM. In the numerical experiments, to test our error estimators (see §4), we use in particular the so-called 1-irregular mesh refinement strategy first proposed in [7] and later detailed in [14]. Since the refinement scheme has been extended to include adaptive finite volume computations, we briefly discuss its basic features.

Recall that a node is called *regular* if it constitutes a vertex for each of the neighboring elements; otherwise it is *irregular*. Figure 2.1 shows a particular 1-irregular mesh where irregular nodes marked by $\times$ are all of index-1 irregularity, that is, the maximum number of irregular nodes on an element side is one. In implementation, no degrees of freedom will be associated with these irregular nodes. Hence, supports of the shape functions defining a basis for an FE space change adaptively with mesh refinements; for example, the shaded subdomains Ω_6, Ω_{17}, and Ω_{21} are the supports of the shape functions corresponding, respectively, to the regular nodes 6, 17, and 21 in Fig. 2.1. The FE spaces so constructed preserve the conformality required by the standard FE approximation provided that some special element constraint methods are used to invoke continuity across interelement boundaries of elements of different size and with shape functions of differing polynomial degree [14].

For finite volume approximation, control volumes have to adapt accordingly to their dual elements. Let $\mathcal{B}$ denote the dual mesh for $\mathcal{T}$. Note that the FVM requires a control volume for each regular node where degrees of freedom are defined. Thus, in particular, 23 control volumes in the dual mesh of Fig. 2.1 are constructed and shown, by dotted lines, in Fig. 2.2. Notice that the pattern of the boundaries of control volumes appearing in elements may differ element by element. This plays an essential role in the implementation since most computations, approximations, as well as error estimations, are to be performed elementwise before the assembling process.

3. Model problems and numerical methods. We now apply the general formulations and the basic ideas discussed above to four model problems in connection with FEM and FVM.

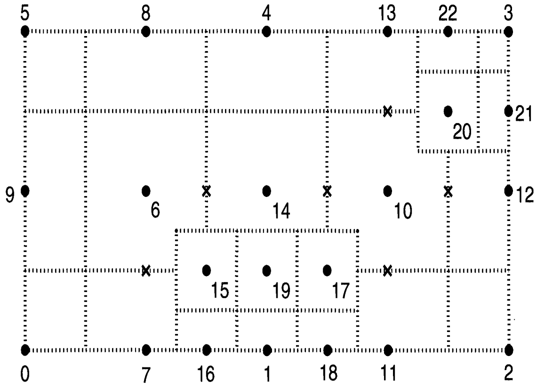


FIG. 2.2. *Adaptive control volumes.*

Since the a posteriori error estimation is the heart of a complete selfadaptive mechanism, we stress particularly various error equations or inequalities on which the error estimators are based. Some rigorous theories of the estimators can be found in [2], [21].

We first note that if the closed convex set K in (2.3) is itself the Sobolev space $H(\Omega)$, then the inequality (2.3) reduces to a *variational equation.* In §§3.1–3.3, 3.5 we will address this equality form.

3.1. Elliptic boundary value problems. Consider the boundary value problem

$$\begin{aligned} &Lu := -\nabla\cdot(a(x)\nabla u(x)) + b(x)u(x) = f(x) \quad \text{in } \Omega, \\ &u = 0 \quad \text{on } \partial\Omega_D, \qquad a(x)\frac{\partial u}{\partial n} = g(x) \text{ on } \partial\Omega_N. \end{aligned} \tag{3.1}$$

The associated variational problem is to find $u \in H(\Omega)$ such that

$$B(u, v) = (f, v) + \langle g, v\rangle_{\partial\Omega_N} \quad \forall v \in H(\Omega), \tag{3.2}$$

where

$$\begin{aligned} H(\Omega) &:= \{u \in H^1(\Omega) : u = 0 \text{ on } \partial\Omega_D\}, \qquad \|\cdot\| := \|\cdot\|_1, \\ B(u, v) &:= \int_\Omega (a\nabla u\nabla v + buv)\,dx, \\ (f, v) &:= \int_\Omega fv\,dx, \qquad \langle g, v\rangle_{\partial\Omega_N} := \int_{\partial\Omega_N} gv\,ds. \end{aligned}$$

To begin with and to avoid technical details, we always assume the unique solvability of the variational problems considered here and below. Note that we do not assume the coefficient function $b(x)$ in (3.1) to be strictly positive in Ω, hence the error estimators to be given apply to indefinite problems as well [21].

Corresponding to (2.4) and (2.7), the approximation and error problems for (3.2) are to determine $u_s \in S$ and $e \in H(\Omega)$ such that

$$B(u_s, v) = (f, v) + \langle g, v\rangle_{\partial\Omega_N} \quad \forall v \in S \tag{3.3}$$

and

$$B(e, v) = -B(u_s, v) + (f, v) + \langle g, v\rangle_{\partial\Omega_N} \quad \forall v \in H(\Omega), \tag{3.4}$$

respectively, where the computed FE solution u_s can only be assessed a posteriori, i.e., after its availability.

Since the space $H(\Omega)$ is still infinite dimensional and the discretization of (3.4) in the original FE space S only produces trivial estimated errors, (3.4) has to be solved in a larger space $\bar{S}$. However, this would cause the error calculation to be impractically expensive since (3.4) will result in a larger system of equations than that of (3.3). Hence, the use of the complementary subspaces appears to be natural. Nevertheless, if the subspace is chosen to be S^c, the error calculation may involve a global solution to a system of equations. We are therefore led to consider the complementary subspaces $S^c_{\mathcal{T}} \subset H_{\mathcal{T}}(\Omega)$ and, consequently, to solve the following reduced problem: Determine $\tilde{e} \in S^c_{\mathcal{T}}(\tau_i)$, in each element $\tau_i \in \mathcal{T}$, such that

$$B(\tilde{e}, v) = -B(u_s, v) + (f, v) + \langle g, v\rangle_{\partial\Omega_N} \quad \forall v \in S^c_{\mathcal{T}}(\tau_i). \tag{3.5}$$

Note that if $S^c_{\mathcal{T}} \subset H(\Omega)$, a conforming subspace, the unique solvability of (3.5) can be ensured as that of (3.3) and only the interior residual in each element will be used. On the other hand, if $S^c_{\mathcal{T}} \subset H_{\mathcal{T}}(\Omega)$ but not in $H(\Omega)$, a nonconforming subspace, then (3.5) results in a nonconforming solution scheme [13] and the estimation will include both interior and boundary residuals for each element. With some moderate assumptions on the bilinear form and the complementary spaces (sufficiently large), it is shown in [21] that the unique solvability for such problems still holds.

As noted already, the error residuals can be expressed in either differential or weak form. We briefly describe the differences between these two approaches. For more detailed theoretical investigation we refer the reader to [11], [21].

Let E be the collection of curves which forms an edge of an element τ in T. The set of edges may be decomposed as the union of two disjoint sets $E = E_B \cup E_I$, where E_B is the set of edges on $\partial\Omega$ and E_I is the set of edges in the interior of Ω. For each edge ε in E, we define a normal direction $n = n_\varepsilon$. More specifically, n_ε is the usual outward normal when $\varepsilon \in E_B$ while, for $\varepsilon \in E_I$, its choice is arbitrary. Let τ_{in}, τ_{out} be two elements sharing an edge ε in E_I and suppose that the normal n is outward from τ_{in}. Then, for x on ε, the jump and the average of v on ε are defined, respectively, by

$$[v]_J(x) = v(x)|_{\tau_{\text{out}}} - v(x)|_{\tau_{\text{in}}} \quad \text{and} \quad [v]_A(x) = \frac{1}{2}\{v(x)|_{\tau_{\text{out}}} + v(x)|_{\tau_{\text{in}}}\}.$$

Substitute $u = e + u_s$ into (3.1) and multiply by a test function v; then we obtain, in each element τ,

$$(Le, v)_\tau = -(Lu_s, v)_\tau + (f, v)_\tau, \tag{3.6}$$

where Lu_s is defined in the sense of distributions. Now integration by parts of the left term in (3.6) yields

$$(Le, v)_\tau = B(e, v)_\tau - \left\langle a\frac{\partial u}{\partial n}, v\right\rangle_{E_\tau} + \left\langle a\frac{\partial u_s}{\partial n}, v\right\rangle_{E_\tau}, \tag{3.7}$$

and after summing (3.6) over all elements, we find that

$$B(e, v) = (f - Lu_s, v) + \left\langle g - a\frac{\partial u_s}{\partial n}, v\right\rangle_{\partial\Omega_N} + \left\langle \left[a\frac{\partial u_s}{\partial n}\right]_J, v\right\rangle_{E_I}. \tag{3.8}$$

This is the standard formulation for estimating errors used by many authors [7], [8], [11], [19]. On the other hand, if the term $(Lu_s, v)_\tau$ in (3.6) is rewritten as in (3.7), then in each element

τ, we obtain

$$B(e, v)_\tau - \left\langle a\frac{\partial u}{\partial n}, v\right\rangle_{E_\tau} = -B(u_s, v)_\tau + (f, v)_\tau.$$

Hence a summation over all elements leads to

$$B(e, v) - \left\langle a\frac{\partial u}{\partial n}, v\right\rangle_{E_B} - \left\langle \left[a\frac{\partial u}{\partial n}\right]_J, [v]_A\right\rangle_{E_I} = -B(u_s, v) + (f, v),$$

which gives (3.4) with more general test functions $v \in H_{\mathcal{T}}(\Omega)$ if the exact solution u is in $H^2(\Omega)$.

It is necessary to reuse the differential operator L when the estimators are calculated based on (3.8). In contrast, in terms of (3.4), the same bilinear formulation (3.2) can be used in both approximation and error estimation. This is certainly preferable, from the user's viewpoint, in adaptive implementation. Moreover, the formal approach can only utilize the complementary spaces $S^c_{\mathcal{T}}$ in a nonconforming setting while the weak residual approach may be used in either a conforming or a nonconforming setting. Finally, in [11], the following saturation condition is introduced for the spaces $\bar{S}$ used in the error analysis:

$$\|u - u_{\bar{s}}\|_1^2 + \left| h_\varepsilon^{1/2}\left[a\frac{\partial(u - u_{\bar{s}})}{\partial n}\right]_J\right|^2_{0,E_I} \le \rho(h)^2\|u - u_s\|_1^2, \tag{3.9}$$

where $u_{\bar{s}}$ is the approximated solution (for analytical purposes only) sought in the larger spaces $\bar{S}$. It is assumed that $\lim_{h\to 0}\rho = 0$; that is, $\bar{S}$ has higher order than S. On the other hand, in weak form, the saturation condition is as follows:

$$\|u - u_{\bar{s}}\|_1 \le \rho\|u - u_s\|_1, \quad 0 \le \rho < 1. \tag{3.10}$$

Thus the condition (3.10) is somewhat weaker than (3.9) since it allows the FE orders of S and $\bar{S}$ to be equal (see, e.g., [21] and Example 4.2 below). The introduction of the jumps in the normal derivative of the computed solutions at interelement boundaries actually forces ρ to become larger and hence deteriorates the quality of the error estimator [21]. Almost all differential-type residual error estimators [3], [4], [7], [8], [11], [19] require some compatibility conditions or auxiliary local problems to overcome this intrinsic difficulty simply because the distribution and jump terms are included in the residual, i.e., the right-hand side of (3.8). These conditions or problems are somewhat ad hoc depending strongly on the model problem under consideration; see the above references. The complementary spaces are essential for both differential and weak residual approaches. With the weak residual form, one can concentrate primarily on the construction of the shape functions of $S^c_{\mathcal{T}}$, which can still handle jumps across element boundaries (with a nonconforming setting), if they are dominant errors.

The norm $|||\tilde{e}|||_{\tau_i} =: \eta_i$, for each element $\tau_i \in \mathcal{T}$, is called the *error indicator* of the element which assesses the quality of the approximate solution u_s in this element and indicates whether the element needs to be refined, derefined, or unchanged. Summing over all elements, the *error estimator* for u_s can be defined by

$$|||\tilde{e}||| = \left[\sum_i \eta_i^2\right]^{1/2}.$$

The error estimator can serve as one of the major stopping criteria for an entire adaptive process. The quality of a proposed error estimator is usually tested by various model problems to which the exact solutions are explicitly known. A computable *effectivity index*

$$\theta = \frac{|||\tilde{e}|||}{|||e|||}$$

is usually introduced to quantify the quality of the estimator and, consequently, the quality of the approximate solution.

3.2. Parametrized nonlinear equations. Stationary problems for many scientific and engineering problems are modeled by a parameter-dependent equation

$$F(u, \lambda) = 0, \tag{3.11}$$

where u is a state variable, λ is a d-dimensional parameter vector, and F denotes some differential operator on a suitable state space. Typically, the solution set $M = F^{-1}(0)$ turns out to be a differentiable submanifold M of dimension d of the product X of the state space and the parameter space. This can be ensured, for instance, when F is a Fredholm map of index d on X [23].

All standard discretizations of a parametrized boundary value problem (3.11) leave the parameter vector untouched and hence approximate the equations by some finite-dimensional system $F_s(u_s, \lambda) = 0$. Hence, under suitable conditions, we may expect the solution set $M_s = F_s^{-1}(0)$ to be a d-dimensional submanifold of some discretization space X_s. Frequently, X_s can be embedded in X and then the discretization error represents some measure of the distance between M and M_s in X.

Our goal now is to estimate the discretization error. There are two major issues for the error estimation of M_s. The first issue is efficiency. It is intrinsically more expensive for nonlinear parametrized problems than it is for linear problems. The second issue is consistency. It is well known [23] that the parameter dependence causes the discretization error to become a local concept which depends on the choice of the local coordinate system on the manifold. For instance, in the continuation method, we must often fix a local coordinate system for calculating several points on the manifold and then change the coordinate system as needed, and so on. However, for error estimation, the coordinate system is usually fixed, [23], throughout the entire manifold and hence is not applicable near any foldpoint with respect to the parameter space. The main way to resolve these difficulties is to use linearization and a local coordinate system.

At any $x_0 = (u_0, \lambda_0) \in M$, we define a local coordinate system that satisfies the following conditions:

$$X = W \oplus T, \quad \dim T = d, \quad W \cap \ker DF(x_0) = \{0\}. \tag{3.12}$$

It is shown in [25] that the constrained linearized (infinite-dimensional) problem

$$F(x_s) + DF(x_s)\omega = 0, \quad \pi(\omega) = 0, \tag{3.13}$$

for (3.11) has a unique solution $\omega = x_0 - x_s \in W$ which is the exact error of the approximate solution $x_s = (u_s, \lambda_s)$. Here $\pi \in L(X)$ is a natural projection of X onto T along W. The FE approximation of (3.11) is of interest here. We consider, in particular, the following mildly nonlinear problem: Find $(u, \lambda) \in H_0^1(\Omega) \times \Lambda$ such that

$$\langle F(u, \lambda), v\rangle := \int_\Omega [a(\lambda, \xi)\nabla u \cdot \nabla v + g(u, \lambda, \xi)v]\, d\xi = 0 \quad \forall v \in H_0^1(\Omega), \tag{3.14}$$

where $F : X = H_0^1(\Omega) \times \Lambda \to Y = H^{-1}(\Omega)$ and the coefficient functions a and g are given so that the problem is well posed.

Apparently, the nonlinear problem (3.14) does not fit into the general variational formulation given in the previous section. However, its weak residual form can be cast in the framework of the variational error setting.

In weak form (3.13) requires the determination of $(w, \mu) \in H_0^1 \times \Lambda$ such that

$$B(w, v) + C(\mu, v) = -\langle F(u_s, \lambda_s), v\rangle \;\; \forall v \in H_0^1(\Omega), \tag{3.15}$$

$$\pi(w, \mu) = 0, \tag{3.16}$$

where

$$B(w, v) := \int_\Omega (a(\lambda_s, \xi)\nabla w \cdot \nabla v + g_u(u_s, \lambda_s, \xi)wv)\, d\xi,$$

$$C(\mu, v) := \int_\Omega ((a_\lambda(\lambda_s, \xi) \cdot \mu)\nabla u_s \cdot \nabla v + (g_\lambda(u_s, \lambda_s, \xi) \cdot \mu)v)\, d\xi.$$

At any computed point $(u_s, \lambda_s) \in M_s$, our a posteriori error estimates thus require the determination of $(\tilde{w}, \tilde{\mu}) \in S_{\mathcal{T}}^c(\tau_i) \times \Lambda$ such that

$$B(\tilde{w}, v) + C(\tilde{\mu}, v) = -\langle F(u_s, \lambda_s), v\rangle \;\; \forall v \in S_{\mathcal{T}}^c(\tau_i), \tag{3.17}$$

$$\pi(\tilde{w}, \tilde{\mu}) = 0. \tag{3.18}$$

The choice of the local coordinate system at any point x_0 on M is arbitrary as long as it satisfies the conditions in (3.12). By definition, the local coordinate system (3.12) and hence the constraint $\pi(\omega, \mu) = 0$ can change from point to point on the solution manifold. We choose $T = \ker DF(x_0)$ in particular. The constraint (3.18) is then equivalent to $(\tilde{w}, \tilde{\mu}) \in W_s = X_s \cap W$ which is orthogonal to the d-dimensional subspace $T = \ker DF_s(x_s)$ of X_s corresponding to the tangent space of M_s at the computed point x_s. If Λ is one dimensional and a standard continuation process is used, then a normalized tangent vector is usually available at each computed point. Analogously, in the multiparameter case, if a triangulation of M_s is computed by the method of [24], then again orthonormal bases of the tangent space are available at the computed points on the manifold. This uniform treatment of the a posteriori error estimation along the computed manifold M_s avoids aforementioned difficulties. Moreover, the introduction of the linear, local, solution scheme ensures that the method is computationally relatively inexpensive.

An a posteriori error estimation has been developed for strongly nonlinear equations with a scalar parameter in [27]. There the asymptotic exactness of a residual estimator was proved under suitable hypotheses. Tsuchiya's approach requires one to fix the local coordinate system in two stages. In the first stage before the turning point, the system is defined by the natural parameter. In the second stage when the continuation process is near and after the turning point, the system is then rotated by 90 degrees thus allowing a more elaborate error estimate near the turning point. In the sense of the definition of the local coordinate system, this is a special case of our approach. Moreover, Tsuchiya's approach requires a global solution to the linearized residual equations for error estimation.

3.3. Symmetric hyperbolic equations. In the theory of PDEs there is a fundamental distinction between those of elliptic, hyperbolic, and parabolic types. The theory of symmetric positive differential equations developed by Friedrichs [17] is known for its unified treatment, analytically as well as numerically, for PDEs that change type within the domain of interest such as the Tricomi problem and forward–backward heat equations. In the development of the error estimator for the Friedrichs system, we use in particular the FEM proposed by Lesaint [20].

An unknown p-dimensional vector-valued function defined on Ω is given by $\mathbf{u} = (u_1, u_2, \dots, u_p)^T$. Let $\mathbf{f} = (f_1, f_2, \dots, f_p)^T$ be a given p-dimensional vector-valued function defined on Ω. Let the operators $\mathbf{L}$ and $\mathbf{M}$ be defined by and consider the following systems:

$$\mathbf{Lu}(x) := \sum_{i=1}^{2} A_i(x)\frac{\partial \mathbf{u}}{\partial x_i} + A_0(x)\mathbf{u} = \mathbf{f}(x) \quad \text{for } x \in \Omega, \tag{3.19}$$

$$\mathbf{Mu}(x) := (\mu(x) - \beta(x))\mathbf{u}(\mathbf{x}) = 0 \quad \text{for } x \in \partial\Omega, \tag{3.20}$$

where $\beta = \sum_{i=1}^{2} n_i A_i$, the n_i's being the components of the outer normal on $\partial\Omega$. The matrices A_i, $i = 1, 2$, are symmetric, Lipschitz continuous in $x = (x_1, x_2)$ for $x \in \bar{\Omega}$. The coefficients of the matrix $A_0(x)$ of $L(R^p)$ are bounded in Ω. The matrix $\mu(x)$ of $L(R^p)$ is defined for $x \in \partial\Omega$ so that the boundary condition (3.20) is *admissible* and the operator $\mathbf{L}$ is *positive* in the sense of Friedrichs [17]; i.e.,

$$\begin{cases} \text{(i)} \;\; \mu(x) + \mu^*(x) \text{ is positive semidefinite on } \partial\Omega, \\ \text{(ii)} \;\; \operatorname{Ker}(\mu - \beta) \oplus \operatorname{Ker}(\mu + \beta) = R^p \text{ on } \partial\Omega, \text{ and} \\ \text{(iii)} \;\; A_0 + A_0^* - \sum_{i=1}^{2} \frac{\partial A_i}{\partial x_i} \geq c_0 I \quad \forall x \in \Omega, \end{cases}$$

where μ^* and A_0^* are adjoint matrices of μ and A_0, respectively, c_0 is a positive constant, and I is the identity matrix.

The adjoint operators $\mathbf{L}^*$ and $\mathbf{M}^*$ of $\mathbf{L}$ and $\mathbf{M}$ are defined, respectively, by

$$\mathbf{L}^*\mathbf{v}(x) := -\sum_{i=1}^{2} \frac{\partial}{\partial x_i}(A_i(x)\mathbf{v}(x)) + A_0^*(x)\mathbf{v}(x) \;\; \forall x \in \Omega, \tag{3.21}$$

$$\mathbf{M}^*\mathbf{v}(x) := (\mu^*(x) + \beta(x))\mathbf{v}(x) \;\; \forall x \in \partial\Omega. \tag{3.22}$$

Let $(\mathbf{q}, \mathbf{g}) := \int_\Omega \mathbf{q}(x) \cdot \mathbf{g}(x)\, dx$ and $\langle \mathbf{q}, \mathbf{g} \rangle := \int_{\partial\Omega} \mathbf{q} \cdot \mathbf{g}\, ds$, where $\mathbf{q} \cdot \mathbf{g} = \sum_{i=1}^{p} q_i g_i$. One variational formulation of (3.19) and (3.20) is to find $\mathbf{u} \in (H^1(\Omega))^p$ such that

$$B(\mathbf{u}, \mathbf{v}) := \frac{1}{2}(\mathbf{Lu}, \mathbf{v}) + \frac{1}{2}(\mathbf{u}, \mathbf{L}^*\mathbf{v}) + \frac{1}{2}\langle \mu\mathbf{u}, \mathbf{v} \rangle = (\mathbf{f}, \mathbf{v}) \;\forall \mathbf{v} \in (H^1(\Omega))^p. \tag{3.23}$$

For any given approximate solution $\mathbf{u}_s \in (S)^p \subset (H^1(\Omega))^p$ of (3.23), again analogous to (2.7), the error estimator (3.23) can then be calculated by solving the reduced error problem: Determine $\tilde{\mathbf{e}} \in (S_T^c)^p$ such that

$$B(\tilde{\mathbf{e}}, \mathbf{v}) = (\mathbf{f}, \mathbf{v}) - B(\mathbf{u}_s, \mathbf{v}) \;\; \forall \mathbf{v} \in (S_T^c)^p. \tag{3.24}$$

Since the boundary conditions (3.20) and (3.22) for test and trial functions $\mathbf{u}$ and $\mathbf{v}$, respectively, are different, this would make it impossible to show the coercivity in a simple manner if the adjoint operator $\mathbf{L}^*$ were not included in (3.23). With the formulation of (3.23) and Friedrichs's identities [17], the coercivity is guaranteed in [20] but only in the lower-order norm, i.e., $\|\cdot\|_0$, instead of the general Gårding-type inequality (2.2).

3.4. Variational inequalities. In the previous subsections, in terms of abstract settings, the closed convex set K is the Sobolev $H(\Omega)$ itself. This in turn leads to variational equations for the preceding model problems. In this subsection, we deal with variational inequalities formulated in the general form (2.3) where now the K is indeed a closed convex subset of $H(\Omega)$. As we would in a typical problem, we shall now also consider the abstract minimization problem: Find $u \in K$ such that

$$J(u) = \inf_{v \in K} J(v), \tag{3.25}$$

where the functional $J : H(\Omega) \to R$ is defined by

$$J(v) = \frac{1}{2}B(v, v) - F(v),$$

provided that the bilinear form $B(\cdot, \cdot)$ is symmetric and $H(\Omega)$-elliptic, i.e., $\alpha = 0$ in (2.2).

Corresponding to (2.4), (3.25) reduces to the finite-dimensional approximate problem

$$J(u_s) = \inf_{v_s \in K_s} J(v_s), \tag{3.26}$$

which, under suitable conditions on the approximate convex set K_s [16], [18] can be solved by mathematical programming subject to a finite number of constraints induced by K_s.

Clearly, the variational error problem (2.7) suggests the needed formulation for error estimation; that is, determine $\tilde{e} \in K_c'$ such that

$$B(\tilde{e}, w) - G(w) \geq B(\tilde{e}, \tilde{e}) - G(\tilde{e}) \quad \forall w \in K_c', \tag{3.27}$$

where K_c' is a complementary closed convex set of $S_{\mathcal{T}}^c$ under similar conditions as those of K_s. Inequality (2.7) also suggests that a new functional can be defined in terms of the weak residual G, namely,

$$E(w) = \frac{1}{2}B(w, w) - G(w).$$

Consequently, we have the following reduced minimization problem: Determine $\tilde{e} \in K_c'$ such that

$$E(\tilde{e}) = \inf_{w \in K_c'} E(w). \tag{3.28}$$

3.5. The finite volume element method. There are many variants of FVMs. We consider specifically the finite volume element method (FVEM). As noted in [12], the FVEM was developed as an attempt to use FE ideas to create a more systematic finite volume (FV) methodology. The basic idea is to approximate the discrete fluxes needed in FV by replacing the unknown PDE solution by an FE approximation. It turns out that the approximate solution by FVEM is in fact sought in a standard FE trial function space whereas the corresponding test function space consists of volumewise constant step functions (zero-order polynomials). In [10], Bank and Rose termed the FVEM the *box method* and showed that, under reasonable hypotheses, the solution u_b generated by the FVEM is of comparable accuracy to the solution u_s generated by the standard Galerkin procedure using piecewise linear FEs. More precisely, the a priori errors of u_b and u_s are of the same order in the energy norm.

As far as a posteriori error estimation is concerned, there are surprisingly fewer results available for the FVMs than for their FE counterparts. From the above observations and by the actual implementation features of the FVEM, we can see that FEMs and FVMs do have many important similarities in error estimation and the adaptive process.

First of all, the FVEM solution u_b is itself an element of the standard FE space S associated with the regular mesh $\mathcal{T}$. Therefore, it is perfectly all right to replace u_s by u_b in (2.6) so that the weak residual G is now in terms of the FV solution u_b. Second, since the solution u_b was computed with degrees of freedom defined at the nodal points of elements instead of volumes, it is quite reasonable to do the error estimation on an element-by-element basis. Third, most computations in practice are customarily carried out elementwise, including control volumes which are constructed according to their dual elements; see, e.g., [10], [12]. Finally, it is interesting to explore how the well-established adaptive features of FE technology can be utilized in FV computations. In short, it is plausible to develop error estimators for the FVEM elementwise instead of volumewise.

We consider again the selfadjoint elliptic boundary value problem (3.1). For the FV element approximation of (3.1), we follow the formulation proposed in [10] which is in a more general setting than that in [12]. The FVEM (or the box method) for (3.1) is defined as follows: Find $u_b \in S$ such that

$$\bar{B}(u_b, \bar{v}) + (b\bar{u}_b, \bar{v}) = (f, \bar{v}) \;\; \forall \bar{v} \in P_0^0(\mathcal{B}), \tag{3.29}$$

where

$$\bar{B}(u_b, \bar{v}) = -\sum_{b_i \in \mathcal{B}} \int_{\partial b_i} a \frac{\partial u_b}{\partial n} \bar{v}\, ds,$$

and $P_0^0(\mathcal{B})$ denotes the space of discontinuous piecewise constants with respect to the control volumes (or boxes), whose elements are zero on $\partial\Omega_D$. Note that to avoid a nondiagonal (and generally nonsymmetric) matrix, $\bar{u}_b$ is used instead of u_b in the second term on the left-hand side of (3.29); see [10]. Here, the $\bar{u}_b$ is defined as a volumewise constant and has the same values as u_b at vertices of $\mathcal{T}$.

From the above observation, we see that the error estimator of the FV solution u_b is proposed by determining $\tilde{e} \in S_{\mathcal{T}}^c(\tau_i)$ such that

$$B(\tilde{e}, v) = -B(u_b, v) + (f, v) + \langle g, v\rangle_{\partial\Omega_N} \quad \forall v \in S_{\mathcal{T}}^c(\tau_i), \tag{3.30}$$

where the bilinear form B is defined exactly as in (3.5).

The FV solution u_b is obtained quite differently in the sense of the abstract setting (2.4). Nevertheless, the error estimation based on (3.30) is still in the unifying theme presented so far. Note also that there is no direct link between (3.29) and (3.30). It is precisely our intention to use the weak residual error estimation. Otherwise, if the boundary integrals $\bar{B}$ were used in (3.30), we would then be forced to perform the estimation volumewise. However, the theoretical investigation of the estimators would certainly involve the a priori results concerning u_b and yet remains open. Equation (3.30) is in exactly the same form as (3.5). As a result, the error indicators and estimators can be calculated in exactly the same way as those of the FEM. Further, the remarks made in §3.1 apply here.

4. Numerical examples. The numerical examples presented in this section correspond with those of the previous subsections. Some examples were also considered in the places cited. Although we intend to stress the performance (the effectivity index) of the respective error estimators, some adaptive computational results are also presented.

There are two different types of the complementary spaces. The conforming shape functions of $S_{\mathcal{T}}^c$ vanish on boundaries of elements; consequently, the error indicators ignore interelement jumps in fluxes. The shape functions for the nonconforming case are discontinuous across element boundaries and hence the error indicators will include both the interior errors and the jumps in, of course, weak form.

Although most applications require only the use of conforming shape functions [1], [9], [25], there are some cases in which this approach would fail (see Example 4.1) if the FE order of S were not properly considered [1], [8]. We shall illustrate both conforming (Examples 4.2, 4.3) and nonconforming (Examples 4.1, 4.4, 4.5) error calculations. General algorithms in implementing these error estimators, for equality-type problems, are detailed in [21], [25]. We shall present an algorithm for the variational inequalities in Example 4.4.

Example 4.1. Following [15] we consider Laplace's equation on an L-shaped domain:

$$\begin{aligned} \Delta u &= 0 \quad \text{in } \Omega = (-1,1) \times (-1,1) \backslash (0,1) \times (-1,0), \\ u &= 0 \quad \text{on } \partial\Omega_D, \qquad \frac{\partial u}{\partial n} = g \quad \text{on } \partial\Omega_N, \end{aligned} \tag{4.1}$$

where $\partial\Omega_D = \{(x, y) : x \in [0, 1],\ y = 0\} \cup \{(x, y) : x = 0,\ y \in [-1, 0]\}$, $\partial\Omega_N = \partial\Omega\backslash\partial\Omega_D$ and g is defined so that, in polar coordinates, the exact solution u becomes $u = r^{2/3} \sin \frac{2\theta}{3}$ and hence has a point singularity at the origin.

For our computations, bilinear elements are used to define S. Hence, by the nature of the problem, the computed solution u_s is a harmonic function in each element and therefore satisfies (4.1) on each element. This means that the errors occur purely on the edges of elements; that is, the complementary space $S^c_{\mathcal{T}}$ has to be nonconforming. For this, we construct four edge midpoint basis functions defined, on the reference element $\hat{\tau} = \{(\xi, \eta) : |\xi| \leq 1,\ |\eta| \leq 1\}$, by

$$(4.2)\qquad \begin{cases} \psi_1(\xi, \eta) = (1 - \xi^2)(1 - \eta)/2, & \psi_2(\xi, \eta) = (1 + \xi)(1 - \eta^2)/2, \\ \psi_3(\xi, \eta) = (1 - \xi^2)(1 + \eta)/2, & \psi_4(\xi, \eta) = (1 - \xi)(1 - \eta^2)/2. \end{cases}$$

For the p-version FEM, this type of construction for the complementary spaces can be readily extended to high-order shape functions. For instance, given basis functions defining S on each element, one can simply add one or more shape functions to the element by higher-order side mode or internal mode or both (see [26]) for $S^c_{\mathcal{T}}$. In essence, this corresponds to a hierarchical basis method [2].

Of course, the conforming-type shape functions can still be used for this kind of problem provided that the original space S is chosen so that the effects of the boundary residuals are negligible. In [8] it was shown that for linear elliptic problems the discretization error of odd-order FE solutions is mainly due to the jumps, while that of even-order approximations occurs principally in the interior of the elements, allowing the jumps across element boundaries to be neglected. In line with this, eight-node biquadratic FEs were used in [1] for the approximation while two fifth-degree polynomials were introduced for the error estimation in weak residual form.

Figure 4.1 is a given initial mesh on the domain. The adaptive computation shown in Fig. 4.2 is very similar to those using the code FEARS (finite element adaptive research solver); see, e.g., [6]. Our approach however is much simpler in terms of implementing the error indicators. Depending on the model problem, the location of singularity, the quantity of interest (displacement, stress, etc.), and the spectral order of the FE, different extraction expressions for an auxiliary problem associated with the model problem should be chosen accordingly for Babuška–Miller indicators; see [6] for more details. Our error indicators are not confined by the above effects; namely, ours do not depend on singularity and spectral order and there are no auxiliary problems. In fact, based on the ideas in the present paper, we are able to develop a very general and robust code which we call **AdaptC++**. So far, we have successfully tested our code for linear elasticity problems, mixed-type problems, flows in porous media, obstacle problems, Navier–Stokes equations, and semiconductor device simulation with FE, FV, and least-squares FE methods. Among other things, one of its advantageous features is the simplicity in implementing the error indicators which disregard input model problems and see only a very general setting of linear and bilinear forms and boundary conditions. From the user's viewpoint, use of the code is even simpler partly due to the refinement scheme and the object-oriented programming language. All numerical data presented in this section except that in Example 4.2 were produced by **AdaptC++**.

Tables 4.1 and 4.2 show that adaptive computations are clearly superior to uniform mesh reductions. For instance, if a tolerance is set to 3% of the relative error (r.e. $= \frac{|||e|||}{|||u|||}$), the uniform approach requires over 10 times the degrees of freedom (DOF) of the adaptive approach. Also, the effectivity indices θ show reliable error estimators for both uniform and adaptive approximations.

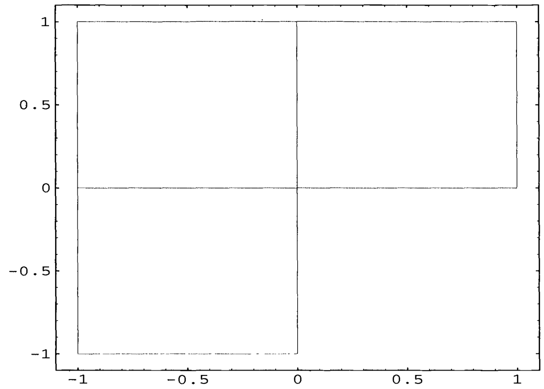


FIG. 4.1. *Initial mesh for Example* 4.1.

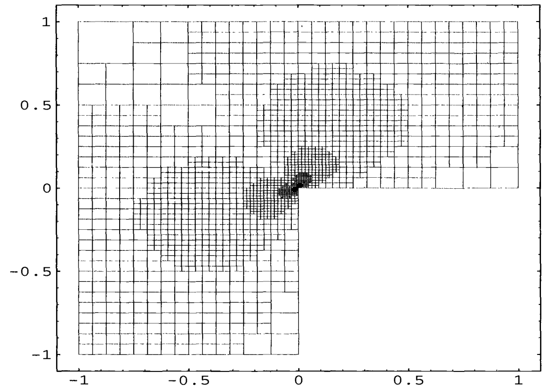


FIG. 4.2. *Adaptive final mesh (FEM) for Example* 4.1.

TABLE 4.1
Example 4.1 *using uniform meshes (FEM).*

DOF	$\|\|\|e\|\|\|$	r.e.	θ
8	0.284368	0.210	0.732
21	0.196695	0.145	0.801
65	0.128699	0.095	0.821
225	0.082777	0.061	0.830
833	0.052781	0.039	0.835
3201	0.033493	0.025	0.837

Example 4.2. The discussions in §3.2 are illustrated by the nonlinear boundary value problem

$$\begin{aligned} &-\Delta u = \lambda e^u, \quad u = u(x,y) \quad \forall (x,y) \in \Omega = (0,1) \times (0,1), \\ &\quad u = 0 \quad \text{on } \partial\Omega. \end{aligned} \tag{4.3}$$

The weak formulation of (4.3) is given as

$$\langle F(u,\lambda), v\rangle := \int_\Omega (u_x v_x + u_y v_y - \lambda e^u v)\, dx\, dy = 0 \;\; \forall v \in H_0^1(\Omega) \tag{4.4}$$

and we assume that $\lambda \in R^1$, which means that the solutions of (4.4) form a one-dimensional manifold M.

TABLE 4.2
Example 4.1 *using adaptive meshes (FEM).*

DOF	$\|\|\|e\|\|\|$	r.e.	θ
8	0.284368	0.210	0.732
21	0.196695	0.145	0.801
34	0.135229	0.100	0.849
53	0.096849	0.071	0.869
78	0.072604	0.053	0.877
119	0.054446	0.040	0.905
301	0.031254	0.023	0.945
469	0.024136	0.017	0.957
765	0.018578	0.013	0.966
1024	0.010675	0.011	0.973
1847	0.011501	0.009	0.980

We use a uniform mesh, $\mathcal{T}$, of 16 biquadratic elements on Ω. For the computation of the one-dimensional solution manifold M_s of the discretized problem, a continuation process (PITCON [23]) is applied starting from $(u_s^0, \lambda_s^0) = (0, 0)$, and our aim is to determine a posteriori estimates of the error between M and M_s at all computed solutions $(u_s, \lambda_s) \in M_s$.

In line with (3.15) and (3.16), the linearized problem at $(u_s, \lambda_s) \in M_s$ is to determine $(w, \mu) \in H_0^1(\Omega) \times \Lambda$ such that

$$\int_\Omega \left[w_x v_x + w_y v_y - \lambda_s e^{u_s} wv + e^{u_s}\mu v\right] dx\,dy \tag{4.5}$$

$$= -\int_\Omega (u_{h_x} v_x + u_{h_y} v_y - \lambda_s e^{u_s} v)\,dx\,dy \quad \forall v \in H_0^1(\Omega),$$

$$\langle (w, \mu), t_s\rangle = 0. \tag{4.6}$$

As noted, the t_s are chosen as normalized tangent vectors on M_s at (u_s, λ_s). Such tangent vectors are available at each step of the continuation process and hence the equation (4.6) involves little additional computational cost.

For local solutions each one, $\tau_i \in \mathcal{T}$, of the 16 elements of Ω is divided into $m = (k+1)^2$ biquadratic subelements with $k = 1, 4$. That means that on each subelement, $\tau_{ij}, i = 1, \dots, 16,\ j = 1, \dots, m$, a bubble-shaped function $\psi_{ij}(x, y)$ is constructed via the mapping of the shape function

$$\psi(\xi, \eta) = (1 - \xi^2)(1 - \eta^2)$$

defined on the reference element. We thus have $S_\mathcal{T}^c(\tau_i) = span\{\psi_{ij}\}_{j=1}^m \subset H_0^1(\tau_i)$ for each element $\tau_i \in \mathcal{T}$ and hence $S_\mathcal{T}^c \subset H_0^1(\Omega)$ is a conforming FE subspace. The more subelements are used, the more accurate auxiliary condition (4.6) becomes and a better quality estimator can be obtained.

The resulting error estimates are shown in Table 4.3, where $|||w_{\bar{s}_k}|||$ denotes the computed error norms for the two cases of k. The computations are very cost-effective, since each local problem involves only a fixed number of degrees of freedom depending on the value of k. The table also shows that, as expected, the estimated errors vary smoothly along the solution path M_s and show no sudden increases near the limit point $\lambda = 6.804524$. As mentioned earlier, if the natural coordinate system induced by the parameter space Λ is chosen, then we expect the resulting error estimates $|||\hat{w}|||$ to become unduly large near the limit point. This is indeed the case as the last column of Table 4.3 shows. At the same time, it should be noted that the computational cost of the two approaches is practically identical.

TABLE 4.3
Example 4.2 *using uniform meshes.*

λ	$\vert\vert\vert w_{\bar{s}_1}\vert\vert\vert$	$\vert\vert\vert w_{\bar{s}_4}\vert\vert\vert$	$\vert\vert\vert\hat{w}\vert\vert\vert$
5.907655	0.017914	0.019054	0.022905
6.193552	0.018368	0.019562	0.024439
6.434373	0.019144	0.020404	0.027792
6.620862	0.020640	0.021986	0.041992
6.745397	0.023317	0.024787	0.159595
6.804524	0.027539	0.029185	0.176171
6.800451	0.033461	0.035345	0.086353
6.740221	0.041066	0.043255	0.069906
6.633252	0.050283	0.052838	0.065171
6.489009	0.061065	0.064041	0.065099
6.328924	0.072480	0.075894	0.067672

Example 4.3. For mixed-type PDEs, we consider the forward–backward heat equation

$$x\phi_t(x,t) - \phi_{xx}(x,t) = f(x,t) \quad \forall (x,t) \in \Omega = (-1,1) \times (0,1), \tag{4.7}$$

$$\begin{cases} \phi(\pm 1, t) = 0 & \forall t \in [0,1], \\ \phi(x,0) = 0 & \forall x \in [0,1], \\ \phi(x,1) = 0 & \forall x \in [-1,0]. \end{cases} \tag{4.8}$$

Note that the equation changes type as x changes sign in Ω. There have been a number of papers addressing this kind of mixed-type heat equation; for further references see [5], [28].

For our computations, the exact solution ϕ is chosen as

$$\phi(x,t) = (x^2-1)t^2[(t-1)^2 - 4x^2] \;\; \forall x \geq 0, \;\; t \in [0,1],$$
$$\phi(x,t) = (x^2-1)(t^2-4x^2)(t-1)^2 \;\; \forall x \leq 0, \;\; t \in [0,1].$$

Denote the boundary $\partial\Omega$ by $\Gamma_1 \cup \cdots \cup \Gamma_6$,

$$\begin{aligned} \Gamma_1 &= \{(x,t) : x \in [-1,0], \;\; t = 0\}, \\ \Gamma_2 &= \{(x,t) : x = -1, \;\; t \in [0,1]\}, \\ \Gamma_3 &= \{(x,t) : x \in [-1,0], \;\; t = 1\}, \\ \Gamma_4 &= \{(x,t) : x \in [0,1], \;\; t = 1\}, \\ \Gamma_5 &= \{(x,t) : x = 1, \;\; t \in [0,1]\}, \\ \Gamma_6 &= \{(x,t) : x \in [0,1], \;\; t = 0\}. \end{aligned}$$

By a change of dependent variables,

$$\mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} e^{-0.1t}\phi \\ e^{-0.1t}\phi_x \end{pmatrix},$$

(4.7) can then be expressed in symmetric positive form

$$L\mathbf{u} := A_1\mathbf{u}_x + A_2\mathbf{u}_t + A_0\mathbf{u} = \mathbf{f}, \tag{4.9}$$

with boundary condition

$$M\mathbf{u} := (\mu - \beta)\mathbf{u} = 0 \quad \forall (x,t) \in \partial\Omega, \tag{4.10}$$

TABLE 4.4
Example 4.3: *Boundary matrices.*

	2μ	2β	M
Γ_1	$\begin{pmatrix} -x & 0 \\ 0 & 0 \end{pmatrix}$	$\begin{pmatrix} -x & 0 \\ 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$
Γ_2	$\begin{pmatrix} 1 & 1 \\ -1 & 0 \end{pmatrix}$	$\begin{pmatrix} -1 & 1 \\ 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ -1 & 0 \end{pmatrix}$
Γ_3	$\begin{pmatrix} -x & 0 \\ 0 & 0 \end{pmatrix}$	$\begin{pmatrix} x & 0 \\ 0 & 0 \end{pmatrix}$	$\begin{pmatrix} -x & 0 \\ 0 & 0 \end{pmatrix}$
Γ_4	$\begin{pmatrix} x & 0 \\ 0 & 0 \end{pmatrix}$	$\begin{pmatrix} x & 0 \\ 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$
Γ_5	$\begin{pmatrix} 1 & -1 \\ 1 & 0 \end{pmatrix}$	$\begin{pmatrix} -1 & -1 \\ -1 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}$
Γ_6	$\begin{pmatrix} x & 0 \\ 0 & 0 \end{pmatrix}$	$\begin{pmatrix} -x & 0 \\ 0 & 0 \end{pmatrix}$	$\begin{pmatrix} x & 0 \\ 0 & 0 \end{pmatrix}$

TABLE 4.5
Example 4.3 *using adaptive meshes.*

DOF	$\|\|\|\mathbf{e}\|\|\|$	r.e.	θ
20	1.310	0.804	1.035
72	0.344	0.211	0.985
156	0.155	0.096	0.972
272	0.087	0.054	0.966
420	0.056	0.034	0.963
600	0.039	0.024	0.961
812	0.029	0.017	0.961

where

$$A_1 = \begin{pmatrix} -x & -1 \\ -1 & 0 \end{pmatrix}, \quad A_2 = \begin{pmatrix} x & 0 \\ 0 & 0 \end{pmatrix}, \quad A_0 = \begin{pmatrix} 0.1x & x \\ 0 & 1 \end{pmatrix}, \quad \mathbf{f} = \begin{pmatrix} e^{-0.1t} f \\ 0 \end{pmatrix},$$

μ, β, and M are given in Table 4.4. It can be readily shown that the system (4.9), (4.10) is symmetric positive.

The adjoint operator $\mathbf{L}^*$ of $\mathbf{L}$ is defined by

$$\mathbf{L}^*\mathbf{v} := \begin{pmatrix} x & 1 \\ 1 & 0 \end{pmatrix} \mathbf{v}_x - \begin{pmatrix} x & 0 \\ 0 & 0 \end{pmatrix} \mathbf{v}_t + \begin{pmatrix} 0.1x + 1 & 0 \\ x & 1 \end{pmatrix} \mathbf{v}.$$

Now, the weak formulation corresponding to (3.23) for (4.7), (4.8) is complete.

For FE approximation of (3.23), a uniform mesh is introduced on Ω and bilinear elements are used. On the other hand, the bubble-shaped functions

$$\psi(\xi, \eta) = \xi(1 - \xi^2)(1/4 - \xi^2)\eta(1 - \eta^2)(1/4 - \eta^2) \tag{4.11}$$

are used to define $(S_{\mathcal{T}}^c)^2$. The effectivity of error estimates using (3.24) is shown in Table 4.5.

In view of error equations (3.5) and (3.24), the calculation of error indicators for the present example proceeds in a similar way as that of Example 4.1 except that we now have, in each element, a 2 × 2 system induced by (3.24) and by the complementary functions (4.11) for the vector-valued functions. However, by the nature of symmetric positive linear differential equations, the bilinear form associated with the system is coercive only in the L^2 norm instead of in the usual H^1 norm for elliptic problems. Hence, there is no equivalence of the energy

TABLE 4.6
Example 4.4 *using adaptive meshes.*

DOF	$\|\|\|e\|\|\|$	r.e.	θ
32	0.0223	0.1930	0.890
92	0.0117	0.1015	0.956
345	0.0053	0.0462	0.958
1489	0.0024	0.0211	0.980
5156	0.0012	0.0108	0.986
6833	0.0011	0.0096	0.987

norm found in elliptic problems. Our numerical experience has shown that the quadratic order of the complementary basis functions used as in Example 4.1 gives rather pessimistic error estimates. The only remedy to this difficulty seems to be to use higher order $(S^2_{\mathcal{T}})^2$, e.g., (4.11). Note that this will not cause more computational cost, in fact, the cost is almost the same as when lower orders are used. Apparently, the a posteriori error analysis for mixed-type PDEs remains largely to be further investigated.

Example 4.4. Our error estimators for variational inequalities (2.3) are tested by the model problem given in [4] with the convex set $K = \{v(x, y) \in H^1(\Omega) : v(x, y) \geq 0$ and $v(x, y) = g(x, y)$ on $\partial\Omega\}$, where $\Omega = (0, \frac{1}{2}) \times (0, 1)$ and g is chosen so that we have the exact solution

$$u = \begin{cases} 0, & \text{if } (x+1)^2 + y^2 \geq 2, \\ \dfrac{1}{4}[(x+1)^2 + y^2] - \dfrac{1}{2} - \dfrac{1}{2}\ln\{[(x+1)^2 + y^2]\}, & \text{otherwise.} \end{cases}$$

The bilinear and linear forms are defined by

$$B(u, v) := \int_\Omega \nabla u \cdot \nabla v \, dxdy, \qquad F(v) := \int_\Omega v \, dxdy.$$

ALGORITHM.

1. Given an initial mesh Ω_h on Ω.
2. Construct a convex set K_s with linear shape functions on Ω_h.
3. Use the Gauss–Seidel–SOR method [18] with the relaxation parameter and the relative error chosen to be 1.2 and 10^{-6}, respectively, to obtain an approximate solution $u_s \in K_s$ of the minimization problem (3.26).
4. Construct the complementary convex set $K'_c := \{w \in S^c_{\mathcal{T}} \subset H_{\mathcal{T}}(\Omega)$ and $w \geq -u_s\}$, where $S^c_{\mathcal{T}}$ is defined via (4.2).
5. In each element τ_i, use the method in Step 3 to solve the reduced problem (3.28) for $\tilde{e} \in K'_c$, which involves only four equations with four unknowns, and then calculate the error indicator η_i for that element. Calculate the error estimator for u_s. If r.e. > 0.01 then refine all elements with $\eta_i \geq 0.1\eta_{\max}$, $\eta_{\max} = \max_i \eta_i$, and go back to Step 2, otherwise stop.

The numerical results are shown in Table 4.6.

Example 4.5. To compare FV and FE computations, we consider again the L-shaped problem (4.1).

For any particular (1-irregular) mesh, e.g., Fig. 4.3, on Ω, the FV element approximation of (4.1) is to find $u_b \in S$ such that

$$-\sum_{b_i \in \mathcal{B}} \int_{\partial b_i} \frac{\partial u_b}{\partial n} ds = 0,$$

where the FE space S is defined as in Example 4.1. Note that the test functions defined on the volumes $b_i \in \mathcal{B}$ are equal to one.

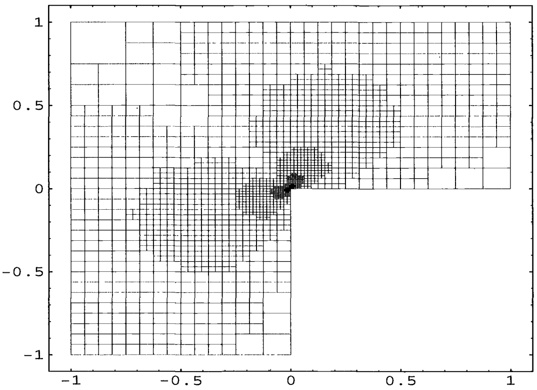


FIG. 4.3. *Adaptive final mesh (FVEM) for Example* 4.5.

TABLE 4.7
Example 4.5 *using uniform meshes (FVEM).*

DOF	$\|\|\|e\|\|\|$	r.e.	θ
8	0.293127	0.217	0.748
21	0.199400	0.147	0.795
65	0.130269	0.096	0.817
225	0.083735	0.062	0.827
833	0.053375	0.039	0.832

TABLE 4.8
Example 4.5 *using adaptive meshes (FVEM).*

DOF	$\|\|\|e\|\|\|$	r.e.	θ
8	0.293127	0.217	0.748
21	0.199400	0.147	0.795
34	0.136928	0.101	0.852
53	0.098177	0.072	0.874
78	0.073441	0.054	0.884
119	0.055017	0.041	0.910
188	0.041540	0.031	0.930
301	0.031497	0.023	0.947
459	0.024512	0.018	0.960
749	0.018927	0.014	0.965
1188	0.014646	0.011	0.973
1801	0.011699	0.009	0.979

Corresponding to Tables 4.1 and 4.2, the data of the FV computations are shown in Tables 4.7 and 4.8, respectively. The final adaptive mesh is shown in Fig. 4.3.

Acknowledgment. The author would like to express his gratitude to the referees for many valuable comments on the paper.

REFERENCES

[1] S. ADJERID AND J. E. FLAHERTY, *Second-order finite element approximations and a posteriori error estimation for two-dimensional parabolic systems*, Numer. Math., 53 (1988), pp. 183–198.

[2] S. ADJERID, J. E. FLAHERTY, AND Y. J. WANG, *A posteriori error estimation with finite element methods of lines for one-dimensional parabolic systems*, Numer. Math., 65 (1993), pp. 1–21.

[3] M. Ainsworth and J. T. Oden, *A unified approach to a posteriori error estimation using element residual methods*, Numer. Math., 65 (1993), pp. 23–50.

[4] M. Ainsworth, J. T. Oden, and C. Y. Lee, *Local a posteriori error estimators for variational inequalities*, Numer. Methods Partial Differential Equations, 9 (1993), pp. 23–33.

[5] A. K. Aziz and J.-L. Liu, *A weighted least squares method for the backward-forward heat equation*, SIAM J. Numer. Anal., 28 (1991), pp. 156–167.

[6] I. Babuška and A. Miller, *The post-processing approach in the finite element methods, Part* 1: *Calculation of displacements, stresses and other higher derivatives of the displacements*, Internat. J. Numer. Methods Engrg., 20 (1984), pp. 1085–1109; *Part* 2: *The calculation of stress intensity factors,* Internat. J. Numer. Methods Engrg., 20 (1984), pp. 1111–1129; *Part* 3: *A posteriori error estimates and adaptive mesh selection*, Internat. J. Numer. Methods Engrg., 20 (1984), pp. 2311–2324.

[7] I. Babuška and W. C. Rheinboldt, *Error estimates for adaptive finite element computations*, SIAM J. Numer. Anal., 15 (1978), pp. 736–754.

[8] I. Babuška and D. Yu, *Asymptotically exact a posteriori error estimator for bi-quadratic elements*, Finite Elements Anal. Design, 3 (1987), pp. 341–354.

[9] P. L. Baehmann, M. S. Shephard, and J. E. Flaherty, *A posteriori error estimation for triangular and tetrahedral quadratic elements using interior residuals*, Internat. J. Numer. Methods Engrg., 34 (1992), pp. 979–996.

[10] R. E. Bank and D. J. Rose, *Some error estimates for the box method*, SIAM J. Numer. Anal., 24 (1987), pp. 777–787.

[11] R. E. Bank and A. Weiser, *Some a posteriori error estimators for elliptic partial differential equations*, Math. Comp., 44 (1985), pp. 283–301.

[12] Z. Cai, *On the finite volume element method*, Numer. Math., 58 (1991), pp. 713–735.

[13] P. G. Ciarlet, *The Finite Element Method for Elliptic Problems*, North–Holland, New York, 1980.

[14] L. Demkowicz, J. T. Oden, W. Rachowicz, and O. Hardy, *Toward a universal h-p adaptive finite element strategy. Part* 1. *Constrained approximation and data structure*, Comput. Methods Appl. Mech. Engrg., 77 (1989), pp. 79–112.

[15] B. Erdmann, R. Roitzsch, and F. Bornemann, *KASADE Numerical experiments*, TR91-1 Konrad-Zuse-Zentrum Berlin (ZIB), 1991.

[16] R. S. Falk, *Error estimates for the approximation of a class of variational inequalities*, Math. Comp., 28 (1974), pp. 963–971.

[17] K. O. Friedrichs, *Symmetric positive differential equations*, Comm. Pure Appl. Math., 11 (1958), pp. 333–418.

[18] R. Glowinski, J. L. Lions, and R. Trémoliéres, *Numerical Analysis of Variational Inequalities*, North–Holland, Amsterdam, 1981.

[19] D. W. Kelly, *The self-equilibration of residuals and complementary a posteriori error estimates in the finite element method*, Internat. J. Numer. Methods Engrg., 20 (1984), pp. 1491–1506.

[20] P. Lesaint, *Finite element methods for symmetric hyperbolic equations*, Numer. Math., 21 (1973), pp. 244–255.

[21] J.-L. Liu and W. C. Rheinboldt, *A posteriori finite element error estimators for indefinite elliptic boundary value problems*, Numer. Funct. Anal. Optim., 15 (1994), pp. 335–356.

[22] J. T. Oden, L. Demkowicz, W. Rachowicz, and T. A. Westermann, *Toward a universal h-p adaptive finite element strategy, Part* 2. *A posteriori error estimation*, Comput. Methods Appl. Mech. Engrg., 77 (1989), pp. 113–180.

[23] W. C. Rheinboldt, *Numerical Analysis of Parametrized Nonlinear Equations*, J. Wiley & Sons, New York, 1986.

[24] ———, *On the computation of multi-dimensional solution manifolds of parametrized equations*, Numer. Math., 53 (1988), pp. 165–181.

[25] W. C. Rheinboldt and J.-L. Liu, *A posteriori error estimates for parametrized nonlinear equations*, in Nonlinear Computational Mechanics, P. Wriggers and W. Wagner, eds., Springer-Verlag, Berlin, 1991, pp. 31–46.

[26] B. A. Sźabo and I. Babuška, *Finite Element Analysis*, Wiley, New York, 1991.

[27] T. Tsuchiya, *A Priori and A Posteriori Error Estimates of Finite Element Solutions of Parametrized Nonlinear Equations*, Ph.D. thesis, Dept. of Math., Univ. of Maryland, College Park, MD, 1990.

[28] V. Vanaja and R. B. Kellogg, *Iterative methods for a forward-backward heat equation*, SIAM J. Numer. Anal., 27 (1990), pp. 622–635.

[29] O. C. Zienkiewicz and J. Z. Zhu, *A simple error estimator and adaptive procedure for practical engineering analysis*, Internat. J. Numer. Methods Engrg., 24 (1987), pp. 337–357.

SIAM J. SCI. COMPUT.
Vol. 17, No. 5, pp. 1269–1274, September 1996

TIMELY COMMUNICATION

Under the "timely communications" policy for the SIAM Journal on Scientific Computing, papers that have significant timely content and do not exceed five pages automatically will be considered for a separate section of the journal with an accelerated reviewing process. It will be possible for the note to appear approximately six months after the date of acceptance.

PRECONDITIONING COMPLICATED FINITE ELEMENTS BY SIMPLE FINITE ELEMENTS[*]

SUSANNE C. BRENNER[†]

Abstract. We discuss a method of preconditioning finite elements with a large number of degrees of freedom by simpler finite elements.

Key words. finite elements, preconditioning, additive Schwarz methods

AMS subject classification. 65N30

1. Introduction.

In recent years the additive Schwarz theory (cf. [12], [17]) has been successfully used in the construction of multilevel preconditioners. In this paper we show that it also provides a useful conceptual framework for the preconditioning of complicated finite elements by simple finite elements on the same level.

It has long been folklore in the finite element community (cf. [2], [8]) that for second-order problems one can use the conforming $\mathcal{P}_1$ element to construct optimal preconditioners for higher-order elements. We give a transparent explanation of this folklore within the additive Schwarz framework in our first example. For fourth-order problems, conforming elements are C^1 elements which are, in general, quite complicated. In our second example we construct optimal preconditioners for such elements using simple nonconforming finite elements. Consequently, multigrid and domain decomposition results for nonconforming finite elements obtained in recent years become relevant even for those who prefer C^1 elements.

For the convenience of the reader, we first give a self-contained treatment of some results (Lemmas 1–3) from the theory of additive Schwarz preconditioners.

Let $\big(V, (\cdot,\cdot)_V\big)$ and $\big(W_j, (\cdot,\cdot)_{W_j}\big)$, $1 \le j \le 2$, be three finite-dimensional inner product spaces. Let $A : V \longrightarrow V$ and $A_j : W_j \longrightarrow W_j$ $(j = 1, 2)$ be linear symmetric positive definite operators. Assume that there exist linear maps $I_1 : W_1 \longrightarrow V$ and $I_2 : W_2 \longrightarrow V$ such that

$$V = I_1(W_1) + I_2(W_2). \tag{1}$$

We shall denote the transpose of I_j by I_j^t; i.e., $(I_j w, v)_V = (w, I_j^t v)_{W_j}$ for all $w \in W_j$ and $v \in V$. Let $B : V \longrightarrow V$ be the preconditioner of A defined by

$$B := I_1 A_1^{-1} I_1^t + I_2 A_2^{-1} I_2^t. \tag{2}$$

LEMMA 1. *The operator B is positive definite with respect to $(\cdot,\cdot)_V$.*

Proof. That B is symmetric positive semidefinite follows immediately from (2). If $(Bv, v)_V = 0$ for some $v \in V$, then $(A_1^{-1} I_1^t v, I_1^t v)_{W_1} + (A_2^{-1} I_2^t v, I_2^t v)_{W_2} = 0$. It follows that

$$I_1^t v = I_2^t v = 0. \tag{3}$$

[*] Received by the editors November 14, 1994; accepted for publication (in revised form) April 1, 1996.

[†] Department of Mathematics, University of South Carolina, Columbia, SC 29208 (brenner@math.sc.edu). This work was supported in part by NSF grant DMS-94-96275.

Let $\tilde{v} \in V$ be arbitrary. By (1) we can write $\tilde{v} = I_1 w_1 + I_2 w_2$ for some $w_1 \in W_1$, $w_2 \in W_2$. Then (3) implies that $(v, \tilde{v})_V = (v, I_1 w_1 + I_2 w_2)_V = (I_1^t v, w_1)_{W_1} + (I_2^t v, w_2)_{W_2} = 0$. Therefore we have $v = 0$, and B is positive definite. □

It follows from Lemma 1 that the operator BA is symmetric positive definite with respect to $(A\cdot, \cdot)_V$ (or $(B^{-1}\cdot, \cdot)_V$), and the eigenvalues of BA are all positive.

LEMMA 2. *The following identity holds:*

$$(4) \qquad (B^{-1}v, v)_V = \inf_{\substack{v = I_1 w_1 + I_2 w_2 \\ w_1 \in W_1, w_2 \in W_2}} \left[(A_1 w_1, w_1)_{W_1} + (A_2 w_2, w_2)_{W_2}\right] \quad \forall v \in V.$$

Proof. Let $v \in V$. Then $v = I_1 w_1 + I_2 w_2$ for some $w_1 \in W_1$ and $w_2 \in W_2$. It follows from the Cauchy–Schwarz inequality and (2) that

$$\begin{aligned}
(B^{-1}v, v)_V &= (B^{-1}v, I_1 w_1)_V + (B^{-1}v, I_2 w_2)_V \\
&= (I_1^t B^{-1} v, w_1)_{W_1} + (I_2^t B^{-1} v, w_2)_{W_2} \\
&= (A_1^{-1/2} I_1^t B^{-1} v, A_1^{1/2} w_1)_{W_1} + (A_2^{-1/2} I_2^t B^{-1} v, A_2^{1/2} w_2)_{W_2} \\
&\le \sqrt{(I_1 A_1^{-1} I_1^t B^{-1} v, B^{-1} v)_V \, (A_1 w_1, w_1)_{W_1}} \\
&\quad + \sqrt{(I_2 A_2^{-1} I_1^t B^{-1} v, B^{-1} v)_V \, (A_2 w_2, w_2)_{W_2}} \\
&\le \sqrt{(v, B^{-1} v)_V} \sqrt{(A_1 w_1, w_1)_{W_1} + (A_2 w_2, w_2)_{W_2}}.
\end{aligned}$$

Therefore we have $(B^{-1}v, v)_V \le \inf\left[(A_1 w_1, w_1)_{W_1} + (A_2 w_2, w_2)_{W_2}\right]$. The opposite inequality follows from the special decomposition where $w_1 = A_1^{-1} I_1^t B^{-1} v$ and $w_2 = A_2^{-1} I_2^t B^{-1} v$. □

Note that Lemma 2 also follows from the so-called fictitious domain lemma (cf. [18], [13]). The following lemma can be traced back to Lions ([15]; cf. also [17] and [12]).

LEMMA 3. *Let C_1 and C_2 be two positive constants such that*

$$(5) \qquad C_1 (Av, v)_V \le \inf_{\substack{v = I_1 w_1 + I_2 w_2 \\ w_1 \in W_1, w_2 \in W_2}} \left[(A_1 w_1, w_1)_{W_1} + (A_2 w_2, w_2)_{W_2}\right] \le C_2 (Av, v)_V$$

for all $v \in V$. Then we have

$$\lambda_{\max}(BA) \le C_1^{-1} \quad \textit{and} \quad \lambda_{\min}(BA) \ge C_2^{-1}.$$

Proof. We have the following identities:

$$(6) \qquad \lambda_{\max}(BA) = \max_{v \ne 0} \frac{(Av, v)_V}{(B^{-1}v, v)_V} \quad \text{and} \quad \lambda_{\min}(BA) = \min_{v \ne 0} \frac{(Av, v)_V}{(B^{-1}v, v)_V}.$$

The lemma follows immediately from (4), (5), and (6). □

In applications we look for W_j, I_j, and A_j such that (i) condition (1) holds, (ii) the constants C_1 and C_2 in (5) are independent of the mesh size, and (iii) the operators I_1, I_2, A_1^{-1}, and A_2^{-1} can be constructed efficiently.

Example 1. Let Ω be a polygonal domain in $\mathbb{R}^2$. We consider the Dirichlet problem for the Laplace equation

$$(7) \qquad -\Delta u = f \quad \text{in } \Omega, \qquad u = 0 \quad \text{on } \partial\Omega.$$

Let $\mathcal{T}_h$ be a quasi-uniform triangulation (cf. [8]) of Ω and V_h be the quadratic finite element space $\{v \in H_0^1(\Omega) : v\big|_T \in \mathcal{P}_2(T)\,\forall\, T \in \mathcal{T}_h\}$. The inner product $(\cdot,\cdot)_{V_h}$ is defined by $(v_1, v_2)_{V_h} := h^2(\sum_p v_1(p)v_2(p) + \sum_m v_1(m)v_2(m))$, where the summation is over all the vertices p and midpoints m. The operator $A : V_h \longrightarrow V_h$ is defined by

$$(8)\qquad (Av_1, v_2)_{V_h} := \int_\Omega \nabla v_1 \cdot \nabla v_2\, dx \quad \forall v_1, v_2 \in V_h.$$

Let W_h be the linear finite element space $\{w \in H_0^1(\Omega) : w\big|_T \in \mathcal{P}_1(T) \text{ for all } T \in \mathcal{T}_h\}$ with the following inner product: $(w_1, w_2)_{W_h} := \sum_p h^2 w_1(p)w_2(p)$. The operator $\tilde{A} : W_h \longrightarrow W_h$ is the analogue of A:

$$(9)\qquad (\tilde{A}w_1, w_2)_{W_h} := \int_\Omega \nabla w_1 \cdot \nabla w_2\, dx \quad \forall w_1, w_2 \in W_h.$$

For $v \in V_h$, let $v^I \in W_h$ be the nodal interpolant of v. We define the subspace $\tilde{V}_h$ of V_h to be $\{v \in V_h : v^I = 0\}$. Let $I_1 : \tilde{V}_h \longrightarrow V_h$ and $I_2 : W_h \longrightarrow V_h$ be natural injections. Clearly we have $V_h = I_1(\tilde{V}_h) \oplus I_2(W_h)$. The preconditioner B of A is defined to be $B := h^2 I_1 I_1^t + I_2 \tilde{A}^{-1} I_2^t$.

LEMMA 4. *There exist positive constants C_1 and C_2, independent of h, such that*

$$(10)\qquad C_1(Av, v)_{V_h} \le h^{-2}(v - v^I, v - v^I)_{V_h} + (\tilde{A}v^I, v^I)_{W_h} \le C_2(Av, v)_{V_h}.$$

Proof. From here on we shall adopt the notation $\lesssim$ and $\approx$. The statement $F \lesssim G$ means that F is bounded by G multiplied by a constant which is independent of h and the functions involved. The statement $F \approx G$ means $F \lesssim G$ and $G \lesssim F$.

It is clear from (8) and (9) that

$$(11)\qquad (Av, v)_{V_h} = |v|^2_{H^1(\Omega)} \quad \forall\, v \in V_h \quad \text{and} \quad (\tilde{A}w, w)_{W_h} = |w|^2_{H^1(\Omega)} \quad \forall\, w \in W_h.$$

By the quasi uniformity of $\mathcal{T}_h$ and a standard homogeneity (scaling) argument we have

$$(12)\qquad (v, v)_{V_h} \approx \|v\|^2_{L^2(\Omega)} \quad \forall\, v \in V_h.$$

We also have the following standard inverse estimate (cf. [8], [10]):

$$(13)\qquad |v|_{H^1(\Omega)} \lesssim h^{-1}\|v\|_{L^2(\Omega)} \quad \forall\, v \in V_h.$$

Then (11)–(13) imply that $(Av, v)_{V_h} = |v|^2_{H^1(\Omega)} \lesssim |v - v^I|^2_{H^1(\Omega)} + |v^I|^2_{H^1(\Omega)} \lesssim h^{-2}\|v - v^I\|^2_{L^2(\Omega)} + |v^I|^2_{H^1(\Omega)} \lesssim h^{-2}(v - v^I, v - v^I)_{V_h} + (\tilde{A}v^I, v^I)_{W_h}$. Therefore the first inequality in (10) holds.

By a standard homogeneity argument we find

$$(14)\qquad \|v - v^I\|_{L^2(\Omega)} + h|v^I|_{H^1(\Omega)} \lesssim h|v|_{H^1(\Omega)} \quad \forall\, v \in V_h.$$

We deduce from (11), (12), and (14) that $h^{-2}(v - v^I, v - v^I)_{V_h} + (\tilde{A}v^I, v^I)_{W_h} \lesssim |v|^2_{H^1(\Omega)} = (Av, v)_{V_h}$. Therefore the second inequality in (10) holds. □

We conclude from Lemmas 3 and 4 (where $V = V_h$, $W_1 = \tilde{V}_h$, $W_2 = W_h$, $A_1 = h^{-2}I$, and $A_2 = \tilde{A}$) that the condition number of BA is bounded by a constant independent of h. We can also define a preconditioner $\tilde{B}$ by $\tilde{B} := h^{-2} I_1 I_1^t + I_2 R I_2^t$, where R is a positive definite operator on $(W_h, (\cdot,\cdot)_{W_h})$ such that $(R^{-1}w, w)_{W_h} \approx (\tilde{A}w, w)_{W_h}\, \forall\, w \in W_h$. For example, R can be a symmetric multigrid approximation of $\tilde{A}^{-1}$ (cf. [3]) or a preconditioner of $\tilde{A}$ based on domain decomposition (cf. [9]). It is clear that (10) holds with different constants (independent of h) if we replace $\tilde{A}$ by R^{-1}. It follows from Lemma 3 that the condition number of $\tilde{B}A$ is also bounded by a constant independent of h. Of course, other higher-order C^0 Lagrange finite element methods for (7) can likewise be preconditioned by the C^0 $\mathcal{P}_1$ element.

Example 2. Let Ω be a polygonal domain in $\mathbb{R}^2$. We consider the Dirichlet problem for the biharmonic equation

$$\Delta^2 u = f \quad \text{in } \Omega, \quad u = \frac{\partial u}{\partial n} = 0 \quad \text{on } \partial\Omega. \tag{15}$$

Let $\mathcal{T}_h$ be a quasi-uniform triangulation of Ω and $V_h = \{v \in H_0^2(\Omega) : v\big|_T \in \mathcal{P}_5(T)\ \forall T \in \mathcal{T}_h$ and the second derivatives of v are continuous at the vertices$\}$ be the quintic Argyris C^1 element (cf. [1]). The inner product on V_h is defined by

$$(v_1, v_2)_{V_h} := h^2 \sum_p \sum_{j=0}^{2} h^{2j} \sum_{|\alpha|=j} \partial^\alpha v_1(p) \partial^\alpha v_2(p) + h^4 \sum_m \frac{\partial v_1}{\partial n}(m) \frac{\partial v_2}{\partial n}(m)$$

for all $v_1, v_2 \in V_h$, where the summation is taken over all the vertices p and midpoints m. The operator $A : V_h \longrightarrow V_h$ is defined by

$$(Av_1, v_2)_{V_h} := \sum_{i,j=1}^{2} \int_\Omega \frac{\partial^2 v_1}{\partial x_i \partial x_j} \frac{\partial^2 v_2}{\partial x_i \partial x_j} dx \quad \forall\, v_1, v_2 \in V_h. \tag{16}$$

Let $W_h = \{w \in L^2(\Omega) : w\big|_T \in \mathcal{P}_2(T)\ \forall\, T \in \mathcal{T}_h$, w is continuous at the vertices and vanishes at the vertices along $\partial\Omega$, $\partial w/\partial n$ is continuous at the midpoints, and $\partial w/\partial n$ vanishes at the midpoints along $\partial\Omega\}$ be the Morley finite element space (c.f. [16]). The inner product on W_h is defined by

$$(w_1, w_2)_{W_h} := h^2 \sum_p w_1(p) w_2(p) + h^4 \sum_m \frac{\partial w_1}{\partial n}(m) \frac{\partial w_2}{\partial n}(m) \quad \forall\, w_1, w_2 \in W_h.$$

The operator $\tilde{A} : W_h \longrightarrow W_h$ is the following nonconforming analogue of A:

$$(\tilde{A} w_1, w_2)_{W_h} := \sum_{T \in \mathcal{T}_h} \left(\int_T \sum_{i,j=1}^{2} \frac{\partial^2 w_1}{\partial x_i \partial x_j} \frac{\partial^2 w_2}{\partial x_i \partial x_j} dx \right) \quad \forall\, w_1, w_2 \in W_h. \tag{17}$$

For each $v \in V_h$, let $v^I \in W_h$ be the nodal interpolant of v. We define the subspace $\tilde{V}_h$ of V_h to be $\{v \in V_h : v^I = 0\}$. Let $J : \tilde{V}_h \longrightarrow V_h$ be the natural injection. We define the map $E : W_h \longrightarrow V_h$ as follows: (i) $(Ew)(p) = w(p)$, (ii) $[\partial(Ew)/\partial n](m) = (\partial w/\partial n)(m)$, (iii) $[\partial^\alpha(Ew)](p) = 0$ for $|\alpha| = 2$, and (iv) $[\partial^\alpha(Ew)](p) =$ average of the values of $(\partial^\alpha w\big|_T)(p)$ over all $T \in \mathcal{T}_h$ which is a subset of the supporting set of the basis functions at p for $|\alpha| = 1$. (Here p and m are the vertices and midpoints of $\mathcal{T}_h$.)

This map E has the following properties (cf. [7]):

$$(Ew)^I = w \quad \forall\, w \in W_h, \tag{18}$$

$$\|w - Ew\|_{L^2(\Omega)} + h^2 |Ew|_{H^2(\Omega)} \lesssim h^2 |w|_{H^2(\mathcal{T}_h)} \quad \forall w \in W_h, \tag{19}$$

where the nonconforming energy norm $|\cdot|_{H^2(\mathcal{T}_h)}$ on the Morley finite element space is defined by $|w|_{H^2(\mathcal{T}_h)} = (\sum_{T \in \mathcal{T}_h} |w|^2_{H^2(T)})^{1/2}$.

It follows from (18) that $V_h = J(\tilde{V}_h) \oplus E(W_h)$. The preconditioner B of A is defined to be $B := h^4 J J^t + E\tilde{A}^{-1}E^t$.

LEMMA 5. *There exist positive constants C_1 and C_2, independent of h, such that*

$$C_1(Av, v)_{V_h} \le h^{-4}(v - Ev^I, v - Ev^I)_{V_h} + (\tilde{A}v^I, v^I)_{W_h} \le C_2(Av, v)_{V_h} \tag{20}$$

for all $v \in V_h$.

Proof. It is clear from (16) and (17) that

$$(Av, v)_{V_h} = |v|^2_{H^2(\Omega)} \quad \forall\, v \in V_h \quad \text{and} \quad (\tilde{A}w, w)_{W_h} = |w|^2_{H^2(\mathcal{T}_h)} \quad \forall\, w \in W_h. \tag{21}$$

By the quasi uniformity of $\mathcal{T}_h$ and a standard homogeneity argument for almost-affine finite elements (cf. [10]) we have

$$(v, v)_{V_h} \approx \|v\|^2_{L^2(\Omega)} \quad \forall\, v \in V_h. \tag{22}$$

The following inequality is a standard inverse estimate (cf. [8], [10]):

$$|v|_{H^2(\Omega)} \lesssim h^{-2}\|v\|_{L^2(\Omega)} \quad \forall\, v \in V_h. \tag{23}$$

The proof of the first inequality in (20) proceeds as in the corresponding proof of Lemma 4 using (19) and (21)–(23).

By the standard homogeneity argument for an almost-affine family of finite elements (cf. [10], [7]) we have

$$\|v - v^I\|_{L^2(\Omega)} + h^2|v^I|_{H^2(\mathcal{T}_h)} \lesssim h^2|v|_{H^2(\Omega)} \quad \forall v \in V_h. \tag{24}$$

We deduce from (19), (21), (22), and (24) that

$$\begin{aligned}
&h^{-4}(v - Ev^I, v - Ev^I)_{V_h} + (\tilde{A}v^I, v^I)_{W_h} \\
&\lesssim h^{-4}\|v - Ev^I\|^2_{L^2(\Omega)} + |v^I|^2_{H^2(\mathcal{T}_h)} \\
&\lesssim h^{-4}\big(\|v - v^I\|^2_{L^2(\Omega)} + \|v^I - Ev^I\|^2_{L^2(\Omega)}\big) + |v|^2_{H^2(\Omega)} \\
&\lesssim |v|^2_{H^2(\Omega)} = (Av, v)_{V_h}.
\end{aligned}$$

Therefore the second inequality in (20) holds. □

It follows from Lemmas 3 and 5 (where $V = V_h$, $W_1 = \tilde{V}_h$, $W_2 = W_h$, $I_1 = J$, $I_2 = E$, $A_1 = h^{-4}I$, and $A_2 = \tilde{A}$) that the condition number of BA is bounded by a constant independent of h. We can also replace $\tilde{A}^{-1}$ by any symmetric positive definite operator $R : W_h \longrightarrow W_h$ such that $(R^{-1}w, w)_{W_h} \approx (\tilde{A}w, w)_{W_h}\, \forall\, w \in W_h$. For example, R can be a symmetric multigrid approximation of $\tilde{A}^{-1}$ (cf. [5], [14]) or a two-level additive Schwarz preconditioner for $\tilde{A}$ (cf. [6], [7]).

Other C^1 elements can also be preconditioned by simpler nonconforming finite elements; for example, the Hsieh–Clough–Tocher macro element (cf. [11]) can be preconditioned by the Morley finite element and the Fraeijs de Veubeke–Sander element (cf. [20]) can be preconditioned by the incomplete biquadratic element (cf. [21]).

Remark. We assume $\mathcal{T}_h$ to be quasi uniform in the two examples only for simplicity. If the mesh size h in the definitions of the inner products and B is replaced by the diameters of the local triangles, our results would hold under the weaker assumption that $\mathcal{T}_h$ is regular.

Remark. We note that results similar to ours have also been obtained independently in the forthcoming papers [4] and [22] and that the opposite approach of preconditioning nonconforming finite elements by conforming ones was investigated in [19].

REFERENCES

[1] J. H. Argyris, I. Fried, and D. W. Scharpf, *The TUBA family of plate elements for the matrix displacement method*, Aero. J. Roy. Aero. Soc., 72 (1968), pp. 701–709.

[2] R. E. Bank and T. F. Dupont, *Analysis of a Two-Level Scheme for Solving Finite Element Equations*, Tech. report CNA-159, University of Texas at Austin, Center for Numerical Analysis, Austin, TX, 1980.

[3] J. H. Bramble, *Multigrid Methods*, Longman Scientific and Technical, Essex, U.K., 1993.

[4] J. H. Bramble, J. E. Pasciak, and X. Zhang, *Two-level preconditioners for 2m'th order elliptic finite element problems*, East-West J. Numer. Math., 1996, to appear.

[5] S. C. Brenner, *An optimal-order nonconforming multigrid method for the biharmonic equation*, SIAM J. Numer. Anal., 26 (1989), pp. 1124–1138.

[6] ———, *Two-level additive Schwarz preconditioners for nonconforming finite elements*, Math. Comp., 65 (1996), pp. 897–921.

[7] ———, *A two-level additive Schwarz preconditioner for nonconforming plate elements*, Numer. Math., 72 (1996), pp. 419–447.

[8] S. C. Brenner and L. R. Scott, *The Mathematical Theory of Finite Element Methods*, Springer-Verlag, New York, Berlin, Heidelberg, 1994.

[9] T. F. Chan and T. P. Mathew, *Domain decomposition algorithms*, in Acta Numerica 1994, A. Iserles, ed., Cambridge University Press, Cambridge, New York, Melbourne, 1994, pp. 61–143.

[10] P. G. Ciarlet, *The Finite Element Method for Elliptic Problems*, North–Holland, Amsterdam, New York, Oxford, 1978.

[11] R. W. Clough and J. L. Tocher, *Finite element stiffness matrices for analysis of plates in bending*, in Proceedings of the Conference on Matrix Methods in Structural Mechanics, Wright Patterson A.F.B., OH, 1965, pp. 515–545.

[12] M. Dryja and O. B. Widlund, *Some Domain Decomposition Algorithms for Elliptic Problems*, Tech. report 438, Department of Computer Science, Courant Institute, New York, 1989.

[13] M. Griebel and P. Oswald, *On the abstract theory of additive and multiplicative Schwarz algorithms*, Numer. Math., 70 (1995), pp. 163–180.

[14] M. R. Hanisch, *Multigrid preconditioning for the biharmonic Dirichlet problem*, SIAM J. Numer. Anal., 30 (1993), pp. 184–214.

[15] P. L. Lions, *On the Schwarz alternating method.* I, in First International Symposium on Domain Decomposition Methods for Partial Differential Equations, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1988, pp. 1–42.

[16] L. S. D. Morley, *The triangular equilibrium problem in the solution of plate bending problems*, Aero. Quart., 19 (1968), pp. 149–169.

[17] S. V. Nepomnyaschikh, *On the application of the bordering method to the mixed boundary value problem for elliptic equations and on mesh norms in $W^{1/2}(S)$*, Soviet J. Numer. Anal. Math. Modelling, 4 (1989), pp. 493–506.

[18] ———, *Fictitious components and subdomain alternating methods*, Soviet J. Numer. Anal. Math. Modelling, 5 (1990), pp. 53–68.

[19] P. Oswald, *Preconditioners for nonconforming elements*, Math. Comp., 65 (1996), pp. 923–941.

[20] G. Sander, *Bornes supérieures et inférieures dans l'analyse matricielle des plaques en flexion-torsion*, Bull. Soc. Roy. Sci. Liège, 33 (1964), pp. 456–494.

[21] Z. Shi, *On the convergence of the incomplete biquadratic nonconforming plate element*, Math. Numer. Sinica, 8 (1986), pp. 53–62. (In Chinese.)

[22] J. Xu, *The auxiliary space method and optimal multigrid preconditioning techniques for unstructured grids*, Computing, 1996, to appear.

SIAM J. SCI. COMPUT.
Vol. 17, No. 6, pp. 1275–1301, November 1996

001

COMPUTING HOPF BIFURCATIONS II: THREE EXAMPLES FROM NEUROPHYSIOLOGY*

JOHN GUCKENHEIMER† AND MARK MYERS‡

Abstract. In [Guckenheimer, Myers, and Sturmfels, *SIAM J. Numer. Anal.*, 34 (1997)] we present algorithms for detecting Hopf bifurcations in two-parameter families of vector fields based on classical algebraic constructions. In addition to their utility as augmented systems for use with standard Newton-type continuation methods, they are shown to be particularly well adapted for solution by computer algebra techniques for vector fields of small or moderate dimension. The present study examines the performance of these methods on test problems selected from models of current research interest in neurophysiology. Implementation issues are examined and the numerical properties of the proposed methods are compared with several alternative algorithms for Hopf pathfollowing that appear in the literature.

Key words. Hopf bifurcation, resultant, bialternate product, neuron model

AMS subject classifications. 58F14, 65H17, 92-08, 92C20

1. Introduction.

The onset of small amplitude oscillations in dynamical systems occurs at Hopf bifurcations. The simplest version of the Hopf theorem is the following.

THEOREM 1.1. *Let*

$$\dot{x} = f(x, \mu), \qquad f : \mathbb{R}^n \times \mathbb{R}^k \to \mathbb{R}^n \tag{1}$$

be a smooth n-dimensional vector field depending upon k parameters with the property that (x_0, μ_0) is an equilibrium point at which the Jacobian matrix $D_x f$ has no zero eigenvalues and a single, simple pair of pure imaginary eigenvalues $\lambda, \bar{\lambda}$. Assume further that $\lambda, \bar{\lambda}$ cross the imaginary axis transversely as the parameters μ are varied. Then there is a smooth submanifold P of dimension $k+1$ containing (x_0, μ_0) that is a union of periodic orbits and equilibrium points of f.

We are concerned with the numerical *detection* of Hopf bifurcation points. This constitutes location of values (x, μ) at which $f(x, \mu) = 0$ and $Df(x, \mu)$ has a pair of pure imaginary eigenvalues. Several methods for computing such points have been proposed and implemented by various authors (for recent reviews, see [18, 24]).

To motivate the discussion that follows, we begin by considering the following planar vector field:

$$\begin{aligned} \dot{x} &= \gamma\left(x + y - x^3/3 + \xi\right), \\ \dot{y} &= -(x - \alpha + \beta y)/\gamma. \end{aligned} \tag{2}$$

This model for the propagation of electrical impulses along a nerve axon was proposed by Fitzhugh [6] as a tractable simplification of the more complicated Hodgkin–Huxley equations discussed as the first example in §2. Equilibrium points for (2) are points which satisfy $\dot{x} = \dot{y} = 0$. Hopf bifurcation points are determined by the Jacobian at the equilibrium. They occur where the trace $\beta/\gamma - \gamma(1 - x^2)$ is zero and the determinant $1 - \beta(1 - x^2)$ is positive. Thus, the determination of Hopf bifurcation points for (2) described above can be written as the following problem:

*Received by the editors August 6, 1993; accepted for publication (in revised form) May 10, 1995. This research was partially supported by grants from the National Science Foundation and the Department of Energy.

†Mathematics Department and Center for Applied Mathematics, Cornell University, Ithaca, NY 14853 (gucken@cam.cornell.edu).

‡Center for Applied Mathematics, Cornell University, Ithaca, NY 14853 (mrm@utrc.utc.com).

Problem (P1):

Find: $(\hat{x}, \hat{y})$ and $(\alpha, \beta, \gamma, \xi)$

Satisfying:

$$\begin{pmatrix} \gamma\left(x + y - x^3/3 + \xi\right) \\ -(x - \alpha + \beta y) \\ \beta/\gamma - \gamma(1 - x^2) \end{pmatrix} = 0$$

Subject To:

$$1 - \beta(1 - x^2) > 0.$$

We seek to extend the approach to detecting Hopf bifurcations in this example to vector fields of higher dimension. We need a nonlinear system of equations comprised of the equilibrium condition of (1) and the algebraic criteria that determine when the Jacobian matrix has a single pair of pure imaginary, conjugate eigenvalues in its spectrum. In [11] we discuss classical algebraic constructions for determining matrices of arbitrary dimension with pure imaginary eigenvalues. A single equality condition and an inequality in the coefficients of the characteristic polynomial are formulated in terms of *resultants*, determinants of special matrices whose elements are functions of the characteristic polynomial coefficients. The equality condition is also derived directly from the Jacobian as the determinant of a *bialternate product* (biproduct) matrix constructed from the Jacobian entries. The biproduct matrix is generally sparse and block structured. Thus, in arbitrary dimensions the nonlinear rootfinding problem corresponding to that of (P1) can be expressed in terms of the defining equations for the vector field and the Jacobian elements alone. We note that there is some overlap in our approach with work of previous authors. In particular, the program *LINLBF* by Khibnik and his coworkers [15] incorporates a method for following curves of Hopf bifurcations based on the resultant of two polynomials which is similar to that used here. In [11] we extend their work by examining several variations of the resultant methods which have differing numerical properties and derive the inequality condition that makes identification of the Hopf points possible. The biproduct formulation is new as is the formulation of conditions under which regularity holds.

Here we discuss the implementation of our new methods, describe their application to three examples from neurophysiology, and make comparisons with other approaches to computing Hopf bifurcations. Each of the examples that we describe comes from a neurophysiological model for the electrical activity of a neuron and each illustrates a different aspect of the methods. In some other problems of modest size or special structure, computer algebra programs produce curves of Hopf bifurcations analytically. However, given an arbitrary nonlinear vector field, solving for the roots of the required system of equations is generally infeasible in closed form. One can attempt to solve (P1) numerically and to follow curves of Hopf bifurcation points in two-parameter families of vector fields. In [11] we show that these augmented systems do, indeed, satisfy the properties required for use in Newton-type numerical continuation.

Our goal in examining these algorithms has been to develop computational methods that facilitate the comparison of biologically based neural models with experimental data. In the first example of §2, we illustrate the application of symbolic methods to the detection of Hopf bifurcations in the classical Hodgkin–Huxley model for action potentials of a squid giant axon. We also use the Hodgkin–Huxley example to compare the accuracy of numerical calculations of derivatives using "automatic differentiation" with finite difference techniques and analytic evaluation. The second example shows the performance of the resultant approach on a more complex model for the electrical activity of a bursting neuron. This model contains a pointof double Hopf bifurcation in its parameter space. We make a detailed comparison of our algorithms with other strategies for detecting Hopf bifurcations for this model. The third example is a still larger and stiffer system of differential equations formulated as a model for electrical activity of a stomatogastric neuron of the crab *Cancer borealis*. We examine the problems caused in the detection of Hopf bifurcations by large eigenvalues in the linearization of this model.

The paper is organized into two sections. The first section is a review of methods for detecting Hopf bifurcations, including those discussed in [11]. The second section presents results obtained for the three examples, beginning with a brief description of the biological origins of each model. We conclude with a summary of our experiments and discuss directions for future work.

2. Review of Hopf bifurcation algorithms.

2.1. Minimally augmented systems. In this section we review strategies for detecting curves of Hopf bifurcation points in two-parameter families of vector fields. We consider first methods based on the algebra of polynomial resultants in families of autonomous vector fields of the form

$$\dot{x} = f(x, \alpha; \beta) \tag{3}$$

where $f : D \subset \mathbb{R}^n \times \mathbb{R} \times I \to \mathbb{R}^n$ and $\beta \in I \subset \mathbb{R}$. For simplicity, we distinguish the second parameter by considering both $x(\beta)$ and $\alpha(\beta)$ to be functions of β. The Jacobian matrix of the vector field is given by

$$\mathbf{J} \equiv \mathbf{J}(x, \alpha; \beta) = \frac{\partial f}{\partial x}(x, \alpha; \beta).$$

$\mathbf{J}$ defines a map from the product space of phase and parameter variables to the space of n-square matrices, denoted by $\mathcal{M}$.

Direct methods for computing curves of Hopf bifurcation for (3) involve appending a determining equation that vanishes when $\mathbf{J}$ has pure imaginary eigenvalues to the equilibrium equations $f(x, \alpha; \beta) = 0$. Thus, we seek a C^2-smooth function $g : \mathbb{R}^n \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}^{n+1}$ so that the *augmented* system

$$F(x, \alpha; \beta) = \begin{pmatrix} f(x, \alpha; \beta) \\ g(x, \alpha; \beta) \end{pmatrix} \tag{4}$$

vanishes at a point of Hopf bifurcation. Furthermore, we require that a point of Hopf bifurcation $(x^*, \alpha^*; \beta^*)$ be a regular solution of (4).

If the characteristic polynomial of $\mathbf{J}$ is given by

$$p(\lambda) = c_0 + c_1\lambda + \cdots + c_{n-1}\lambda^{n-1} + \lambda^n,$$

then p has the nonzero root pair $\{\lambda, -\lambda\}$ if and only if λ is a common root of the two equations $p(\lambda) + p(-\lambda)$ and $p(\lambda) - p(-\lambda)$. Making the substitution $z = \lambda^2$ and rearranging, we construct two new polynomials. If n is even, let

$$\begin{aligned} r_e(z) &= c_0 + c_2 z + c_4 z^2 + \cdots + c_{n-2} z^{\frac{n-2}{2}} + z^{\frac{n}{2}}, \\ r_o(z) &= c_1 + c_3 z + c_5 z^2 + \cdots + c_{n-1} z^{\frac{n-2}{2}}, \end{aligned} \tag{5a}$$

while if n is odd, set

$$\begin{aligned} r_e(z) &= c_0 + c_2 z + c_4 z^2 + \cdots + c_{n-3} z^{\frac{n-3}{2}} + c_{n-1} z^{\frac{n-1}{2}}, \\ r_o(z) &= c_1 + c_3 z + c_5 z^2 + \cdots + c_{n-2} z^{\frac{n-3}{2}} + z^{\frac{n-1}{2}}. \end{aligned} \tag{5b}$$

Then p has a nonzero root pair $\{\lambda, -\lambda\}$ if there exists a z that satisfies

$$\begin{pmatrix} r_e(z) \\ r_o(z) \end{pmatrix} = 0.$$

Two polynomials have a common root if and only if they share a common factor. There are several equivalent ways of determining whether two univariate polynomials have a common root. First, the Euclidean algorithm yields a sequence of polynomials of decreasing degree that are ideally generated by $r_e(z)$ and $r_o(z)$. The last term in this sequence can be expressed as a determinant constructed from the coefficients of two polynomials. We describe one way to do so.

The *Bezout* resultant is a determinant that indicates whether r_e and r_o have a common root. For n even, consider the two polynomials specified by (5a). We define the brackets

$$[i, j] = \det \begin{bmatrix} c_{2i} & c_{2j} \\ c_{2i+1} & c_{2j+1} \end{bmatrix}$$

where $0 \le i, j \le \frac{n}{2}$ and we take $c_n = 1,\ c_{n+1} = 0$. The *Bezout* matrix $\mathcal{B}$ corresponding to the polynomial pair (r_e, r_o) is an $\frac{n}{2}$-dimensional square, symmetric matrix with entries constructed as sums of brackets in the coefficients c_i as follows: for $1 \le i \le j \le \frac{n}{2}$ set

$$\begin{aligned} k_{min} &= \max(0, i + j - n/2 - 1), \\ k_{max} &= i - 1, \end{aligned} \tag{6}$$

$$(\mathcal{B})_{ij} = \sum_{k=k_{min}}^{k_{max}} [i + j - k - 1, k] = (\mathcal{B})_{ji}. \tag{7}$$

The only modification required in this definition for the case of n odd is that c_n has the value prescribed by the characteristic polynomial p and c_{n+1} is taken to be unity. We also define the *Bezout subresultants* $\mathcal{B}_0$ and $\mathcal{B}_1$ as the determinants of the matrices obtained from $\mathcal{B}$ by deleting the first column and the ith row of $\mathcal{B}$. The following theorem is proved in [11].

THEOREM 2.1. *Let B be the Bezout matrix for the polynomials r_e and r_o in* (2). *Then* **J** *has precisely one pair of pure imaginary eigenvalues if*

$$\det(\mathcal{B}) = 0 \quad \textit{and} \quad \det(\mathcal{B}_0) \cdot \det(\mathcal{B}_1) > 0.$$

If $\det(\mathcal{B}) \neq 0$ *or* $\det(\mathcal{B}_0) \cdot \det(\mathcal{B}_1) < 0$, *then* $p(\lambda)$ *has no pure imaginary roots.*

Table 1 provides a list of the resultant equality and subresultant inequality conditions, as functions of the polynomial coefficients, for vector fields of dimensions two to six.

To circumvent possible difficulties in explicitly determining the characteristic polynomial coefficients of **J**, we described a method for determining whether a square matrix has a pair of eigenvalues with zero sum directly from the entries in the Jacobian matrix.

DEFINITION. *Let* **A** *and* **B** *be $n \times n$ matrices with entries (a_{ij}) and (b_{ij}), respectively,* $1 \le i, j \le n$. *Set $m = \frac{n}{2}(n-1)$. Then the* bialternate product *(or biproduct) of* **A** *and* **B**, *denoted*

TABLE 1
Resultant equality and subresultant inequality conditions required for Hopf bifurcation for vector fields of dimensions two through six.

n	$\det(\mathcal{B})$ $\det(\mathcal{S}_0)\cdot\det(\mathcal{S}_1)$
2	c_1
	c_0
3	$c_0 - c_1c_2$
	c_1
4	$c_0c_3^2 - c_1c_2c_3 + c_1^2$
	c_1c_3
5	$(c_2 - c_3c_4)(c_1c_2 - c_0c_3) + c_1c_4(c_1c_4 - 2c_0) + c_0^2$
	$(c_2 - c_3c_4)\cdot(c_0 - c_1c_4)$
6	$c_0c_5^2(c_0c_5 - c_2c_3) + c_1c_5^2(c_2^2 - c_0c_4) + c_1(c_1^2 + c_0c_3c_5)$ $+c_1c_5(c_0c_3 - 2c_1c_2) + (c_4c_5 - c_3)(c_0c_3^2 - c_0c_1c_5 + c_1^2c_4 - c_1c_2c_3)$
	$(c_1c_3 + c_0c_5^2 - c_1c_4c_5)\ \cdot(c_3^2 - c_1c_5 + c_2c_5^2 - c_3c_4c_5)$

$\mathbf{A}\odot\mathbf{B}$, *is an* $m\times m$ *matrix whose rows are labeled* pq ($p = 2, 3, \dots, n$; $q = 1, 2, \dots, p-1$) *and columns labeled* rs ($r = 2, 3, \dots, n$; $s = 1, 2, \dots, r-1$) *with entries*

$$(\mathbf{A}\odot\mathbf{B})_{\{pq,rs\}} = \frac{1}{2}\left\{\begin{vmatrix} a_{pr} & a_{ps} \\ b_{qr} & b_{qs}\end{vmatrix} + \begin{vmatrix} b_{pr} & b_{ps} \\ a_{qr} & a_{qs}\end{vmatrix}\right\}.$$

THEOREM 2.2. *Let* $\mathbf{A}$ *be an* $(n\times n)$ *matrix with eigenvalues* $(\lambda_1, \dots, \lambda_n)$. *Then*

(i) $\mathbf{A}\odot\mathbf{A}$ *has eigenvalues* $\lambda_i\cdot\lambda_j$ *and*

(ii) $2\mathbf{A}\odot\mathbf{I}_n$ *has eigenvalues* $\lambda_i+\lambda_j$

where $\mathbf{I}_n$ *is the n-square identity matrix and* $1\le j<i\le n$.

Substituting $\mathbf{I}_n$ into the definition of the bialternate product and solving for the elements yields a simple formula for the entries. For the $\frac{n}{2}(n-1)$-square matrix $2\mathbf{A}\odot\mathbf{I}_n$ with rows pq and columns rs the entries are given by the formula

$$(8)\qquad (2\mathbf{A}\odot\mathbf{I}_n)_{\{pq,rs\}} = \begin{cases} -(A)_{ps} & \text{if } r = q, \\ (A)_{pr} & \text{if } r \neq p \text{ and } s = q, \\ (A)_{pp} + (A)_{qq} & \text{if } r = p \text{ and } s = q, \\ (A)_{qs} & \text{if } r = p \text{ and } s \neq q, \\ -(A)_{qr} & \text{if } s = p, \\ 0 & \text{otherwise.}\end{cases}$$

From the algebraic theory of symmetric matrix products described above we have a simple *necessary* condition for a Hopf point: if the point (x^*, λ^*) is a Hopf bifurcation point for $\dot{x} = f(x, \alpha)$, then the $(n + 1)$-dimensional system

$$F(x^*, \alpha^*) = \begin{pmatrix} f(x^*, \alpha^*) \\ \det\left(D_x f \odot \mathbf{I}_n\Big|_{(x^*,\alpha^*)}\right) \end{pmatrix} \tag{9}$$

vanishes. However, we have not found a condition that distinguishes pure imaginary eigenvalues from real pairs with zero sum directly from the Jacobian and its bialternate products in analogy to the subresultant criteria described earlier.

Both the determinant of the biproduct and the resultant provide augmentation functions g that can be used in detecting Hopf bifurcations in one-parameter families of vector fields or in applying continuation methods for the computation of curves of Hopf bifurcations in two-parameter families. The regularity theorem of [11] guarantees that the matrix of partial derivatives of the augmented system $F = 0$ has maximum rank $n + 1$ at points of simple Hopf bifurcation. In §2, we discuss several aspects of standard rootfinding and continuation methods that can be improved by taking into account the nature of the augmenting function g.

2.2. Other methods based on the characteristic polynomial. Kubiček has previously described two direct methods for computing Hopf bifurcation points which explicitly require the coefficients of a matrix characteristic polynomial [16, 17]. Both methods result in $(n+2)$-dimensional systems of algebraic equations which may be solved by conventional continuation techniques. Performance of these algorithms has been evaluated on a variety of testcases, including panel flutter and nonadiabatic tubular reactions [24], continuous-stirred tank reactions [13], and parabolic reaction-diffusion equations [23] with favorable results.

The two methods of Kubiček are based on the direct determination of the pure imaginary eigenvalues, say $\lambda^*_{1,2} = \pm\sqrt{\omega} i$, of the Jacobian characteristic polynomial p. Suppose such an eigenvalue pair exists. Then there is a polynomial q of degree $(n-2)$ such that

$$\begin{aligned} p(\lambda) &= (\lambda^2 + \omega) q(\lambda) \\ &= (\lambda^2 + \omega) \sum_{k=0}^{n-2} b_k \lambda^{n-k-2} + A\lambda + B \end{aligned}$$

where the b_i are given recursively by the formula $b_i = c_i - \omega b_{i-2}$ for $1 \leq i \leq n-2$ with $b_{-1} = 0$. We require the constant and linear coefficients A and B to be zero; that is,

$$\begin{pmatrix} A(x, \alpha) \\ B(x, \alpha) \end{pmatrix} = \begin{pmatrix} c_{n-1} - \omega b_{n-3} \\ c_n - \omega b_{n-2} \end{pmatrix} = 0 \tag{10}$$

where the dependence of A and B on (x, α) has been emphasized. Thus, (10) yields a two-dimensional augmented system which must vanish at a Hopf bifurcation point

$$F(x, \alpha, \omega) = \begin{pmatrix} f(x, \alpha) \\ c_{n-1} - \omega b_{n-3} \\ c_n - \omega b_{n-2} \end{pmatrix}. \tag{K1}$$

Kubiček's second method depends on the observation that if (x^*, α^*) is a point of Hopf bifurcation with imaginary eigenvalues $\pm\sqrt{\omega} i$, then the matrix $\mathbf{J}^2$ has a real eigenvalue $-\omega$ with multiplicity equal to two. Suppose the characteristic polynomial of $\mathbf{J}^2$ has r eigenvalues distinct from $-\omega$; that is, we take the spectrum of $\mathbf{J}^2$ to be $\{-\omega, \lambda_1, \ldots, \lambda_r\}$ with respective multiplicities $\{m_\omega, m_1, \ldots, m_r\}$ where $(m_\omega + m_1 + \ldots m_r) = n$. Then q may be written

$$q(\lambda) = (\lambda + \omega)^{m_\omega} \prod_{k=0}^{r} (\lambda - \lambda_k)^{m_k}. \tag{11}$$

Differentiating with respect to λ we obtain

$$\frac{dq}{d\lambda} = m_\omega (\lambda + \omega)^{m_\omega - 1} \prod_{k=0}^{r} (\lambda - \lambda_k)^{m_k} + (\lambda + \omega)^{m_\omega} \frac{d}{d\lambda} \prod_{k=0}^{r} (\lambda - \lambda_k)^{m_k}.$$

For $-\omega$ to be a double real root,

$$q(-\omega) = 0 = \frac{dq}{d\omega}(-\omega),$$

which leads to the augmented system

$$F(x, \alpha, \omega) = \begin{pmatrix} f(x, \alpha) \\ q(-\omega) \\ \frac{dq}{d\omega}(-\omega) \end{pmatrix}. \tag{K2}$$

2.3. Other methods not based on the characteristic polynomial. Here we review methods for computing Hopf bifurcations, which do not require the coefficients of the Jacobian characteristic polynomial. Most of the methods in this category use variants of two distinct defining equations. Our discussion summarizes the main features of these methods; for more details see Roose and Hlavacek [22].

Suppose (x^*, α^*) is a Hopf bifurcation point for (2) with pure imaginary eigenvalues $\pm\omega i$ and associated eigenvector $u + vi$. Denote $E = span(u, v)$, which is two-dimensional if $\mathbf{J}$ is nonsingular. The defining relation

$$(D_x f - \omega \mathbf{I})\,(u + vi) = 0$$

leads to the augmented system

$$F_1(x, \alpha, u, v, \omega) = \begin{pmatrix} f(x, \alpha) \\ [D_x f]\, u - \omega v \\ [D_x f]\, v + \omega u \\ \mathcal{N}(u, v) \end{pmatrix}.$$

If the same construction is applied to the matrix $\mathbf{J}^2 = [D_x f]^2$, we obtain the inflation

$$F_2(x, \alpha, u, \omega) = \begin{pmatrix} f(x, \alpha) \\ [D_x f]^2\, u + \omega^2 u \\ \mathcal{N}(u) \end{pmatrix}.$$

The operator $\mathcal{N} : X \to \mathbb{R}^2$, where X is either $\mathbb{R}^n \times \mathbb{R}^n$ or $\mathbb{R}^n$, is used to normalize the vectors u and v and typically requires a fixed vector l. Precisely how l is selected is a matter of implementation, but it must be chosen so that $l \notin E$. Two choices for $\mathcal{N}$ have been discussed extensively in the literature. Griewank and Reddien [9] suggested the use of

$$\mathcal{N}_1 = \begin{pmatrix} \langle l, u \rangle \\ \langle l, v \rangle - 1 \end{pmatrix}$$

in conjunction with F_1 and showed that both Hopf and simple quadratic turning points are regular, isolated solutions to the system of equations. An alternative proposed by Roose and Hlavacek [22] is to take

$$\mathcal{N}_2 = \begin{pmatrix} \langle l, u \rangle \\ \langle v, v \rangle - 1 \end{pmatrix},$$

which, for l chosen as prescribed above, does not admit turning points as isolated solutions.

Each of the four possible choices available by this construction can be used with standard continuation techniques to track curves of Hopf bifurcation in two-parameter families. For example, if we set $y = (x, \alpha, u, v, \omega)$, then the kth Newton step taken to find a local root $F_1(y) = 0$ will require the solution of the linearized system

$$y_{k+1} = y_k - \left[D_y F_1(y_k)\right]^{-1} F_1(y_k).$$

While it is possible to ignore the special structure of $D_y F(y_k)$ and solve for the updated step with LU decomposition, much faster algorithms may be obtained by exploiting the block structure of the Jacobian for the augmented system. Computer software designed to implement general-purpose numerical bifurcation techniques must be specially modified to accommodate these special Newton-step solution methods. Table 2 summarizes the defining equations for the direct methods described above.

3. Three examples.

3.1. Hodgkin–Huxley equations.

The space-clamped Hodgkin–Huxley equations are a system of four nonlinear ordinary differential equations that describe the electrical response of the giant nerve axon from the squid *Loligo* to an externally applied current [14]. The typical response of the axon to a step in the stimulus current I is characterized by an abrupt spike in the electrical potential difference v between the intracellular fluid and the extracellular medium called an *action potential.* In the Hodgkin–Huxley model this depolarization is induced primarily by an inward flux of sodium (Na^+) followed by an outward flow of potassium (K^+) ions. Other ions contribute to a "leak" current across the axon membrane. The sodium and potassium currents are controlled by three *gating* variables denoted m, n, and h, together with parameters $\bar{g}_{Na}$, $\bar{g}_K$, and $\bar{g}_l$ that measure the maximum conductances of the channels. The resulting vector field is given by

$$\begin{aligned} \dot{v} &= -G(v, m, n, h) - I, \\ \dot{m} &= \Phi(T)\left[(1 - m)\alpha_m(v) - m\beta_m(v)\right], \\ \dot{n} &= \Phi(T)\left[(1 - n)\alpha_n(v) - n\beta_n(v)\right], \\ \dot{h} &= \Phi(T)\left[(1 - h)\alpha_h(v) - h\beta_h(v)\right] \end{aligned} \tag{12}$$

where $\dot{x}$ stands for dx/dt and Φ is given by $\Phi(T) = 3^{(T-6.3)/10}$. The other functions involved are

$$G(v, m, n, h) = \bar{g}_{\mathrm{na}} m^3 h(v - \bar{v}_{\mathrm{na}}) + \bar{g}_{\mathrm{k}} n^4 (v - \bar{v}_{\mathrm{k}}) + \bar{g}_{\mathrm{l}}(v - \bar{v}_{\mathrm{l}})$$

TABLE 2

Summary of the defining equations for the direct methods discussed in the paper for use with pathfollowing numerical codes. Labels are used in the sequel to distinguish the various algorithms.

Label	$g : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}^l$	m	l
RS,RB	$g(x, \alpha; \beta) = \det(\mathcal{B})(x, \alpha; \beta)$	0	1
BP	$g(x, \alpha; \beta) = \det([D_x f] \odot I_n)$	0	1
JGR	$g(x, v, w, \omega, \alpha; \beta) = \begin{pmatrix} [D_x f]\, v - \omega w \\ [D_x f]\, w + \omega v \\ < v, w > -1 \\ < l, v > \end{pmatrix}$	$2n+1$	$2n+2$
RH	$g(x, w, \omega, \alpha; \beta) = \begin{pmatrix} [D_x f]^2\, w - \omega^2 w \\ < w, w > -1 \\ < l, w > \end{pmatrix}$	$n+1$	$n+2$
K1	$g(x, \omega, \alpha; \beta) = \begin{pmatrix} c_{n-1} - \omega b_{n-3} \\ c_n - \omega b_{n-2} \end{pmatrix}$	1	2
K2	$g(x, \omega, \alpha; \beta) = \begin{pmatrix} q(-\omega) \\ \frac{dq}{d\omega}(-\omega) \end{pmatrix}$	1	2

and the following equations modeling the variation of membrane permeability:

$$\begin{aligned} \alpha_m(v) &= \Psi((v+25)/10), & \beta_m(v) &= 4e^{v/18}, \\ \alpha_n(v) &= \tfrac{1}{10}\Psi((v+10)/10), & \beta_n(v) &= 0.125e^{v/80}, \\ \alpha_h(v) &= 0.07e^{v/20}, & \beta_h(v) &= \left(1 + e^{(v+30)/10}\right)^{-1}, \\ \Psi(x) &= x/(e^x - 1). \end{aligned}$$

In the following, we use the temperature $T = 6.3$°C and parameter values for $\bar{g}_{\text{ion}}$ and $\bar{v}_{\text{ion}}$ used by Hodgkin and Huxley [14]:

$$\begin{aligned} \bar{g}_{\text{na}} &= 120\text{mS/cm}^2, & \bar{g}_{\text{k}} &= 36\text{mS/cm}^2, & \bar{g}_{\text{l}} &= 0.3\text{mS/cm}^2, \\ \bar{v}_{\text{na}} &= -115\text{mV}, & \bar{v}_{\text{k}} &= 12\text{mV}, & \bar{v}_{\text{l}} &= 10.599\text{mV}. \end{aligned}$$

The space-clamped Hodgkin–Huxley equations exhibit periodic solutions that arise through Hopf bifurcation with varying (v_k, I) [21]. In general, one does not expect to be able to find explicit analytic expressions for the solutions to multi-dimensional systems of equations, but the special structure of the Hopf bifurcation points allows us to do so. All the Hopf data reported in the discussion which follows was computed using the Maple© symbolic algebra package, reproduced using the same approach in Mathematica© and verified by standard numerical continuation using the augmented systems denoted by (JGR) and (RH) in Table 2 and described in §3.2.

The Jacobian matrix corresponding to (12), tedious to derive by hand, is found easily in Maple©. The corresponding characteristic polynomial

$$p(\lambda) = \lambda^4 + \sum_{i=0}^{3} c_i(v, m, n, h;\, v_k, I) \cdot \lambda^i$$

may be formed and the expressions required by Theorem 2.1 follow by inspection:

$$\det(\mathcal{B}) = c_0 c_3^2 - c_1 c_2 c_3 + c_1^2, \tag{13a}$$

$$\det(\mathcal{B}_0) \cdot \det(\mathcal{B}_1) = c_1 \cdot c_3. \tag{13b}$$

The locus of Hopf bifurcation points in the (v_k, I) plane will be computed as a curve parameterized by v^*, the equilibrium values of v. Suppose that, for a prescribed v_k and I, the point (v^*, m^*, n^*, h^*) is an equilibrium for (12). Since the equations for $\dot{m}$, $\dot{n}$, and $\dot{h}$ depend linearly on m, n, and h,

$$\begin{aligned} m^* &= \frac{\alpha_m(v^*)}{\alpha_m(v^*) + \beta_m(v^*)}, \\ n^* &= \frac{\alpha_n(v^*)}{\alpha_n(v^*) + \beta_n(v^*)}, \\ h^* &= \frac{\alpha_h(v^*)}{\alpha_h(v^*) + \beta_h(v^*)}. \end{aligned} \tag{14}$$

Observe that (14) is independent of I and v_k and have nonzero denominators. Therefore, we obtain equilibrium values of m^*, n^*, and h^* to be used in calculating I and v_k. The Jacobian of (12) is independent of I and (13a) depends quadratically on v_k. Therefore, we can solve (13a) for values $v_k^{\pm}(v^*)$ yielding Hopf bifurcation at the equilibrium (v^*, m^*, n^*, h^*). Finally, substitution of the equilibrium coordinates and the values $v_k^{\pm}(v^*)$ into (12) gives the value of $I = -G$ at the point of Hopf bifurcation. Algorithm 1 summarizes the steps required to compute Hopf bifurcation points in the (v_k, I) parameter plane.

ALGORITHM 1. *Procedure for computing Hopf bifurcation points in Maple© given a value for the external current I.*

```
procedure hh_hopf
    3.0  Choose $v^*$.
    3.1  Evaluate $m^*$, $n^*$, and $h^*$ symbolically.
    3.2  Substitute $m^*$, $n^*$, and $h^*$ expressions into Hopf
         determining conditions and simplify to form $v_k^{\pm}(v^*)$.
    3.3  Check the sign of the subresultant condition.
    3.4  Evaluate $G^{\pm}(v^*)$ and solve for $I = -G(v^*, m^*, n^*, h^*)$.
return $\{v_i^*, m_i^*, n_i^*, h_i^*\}$
```

For nominal values of the other system parameters, there is a single, connected curve of Hopf bifurcation points in the rectangular region [−45, 130] mV × [−150, 400] mA of the (v_k, I) plane. Table 3 lists the phase and parameter space coordinates of 19 selected points used for comparison in the discussion that follows. In particular, points 1 and 19 are chosen near Takens–Bogdanov bifurcations which lie at either terminus of the Hopf

TABLE 3

Phase- and parameter-space coordinates for the selected points in the Hopf curve. Points 1 *through* 10 *lie near a Takens–Bogdanov point at one end of the Hopf curve. The last column shows the computed magnitude of the pure imaginary eigenvalue.*

Pt	v	m	n	h	v_k	I	Im(λ)
1	−23.072	0.44955	0.65670	0.06213	−30.34	−100.80	0.0097
2	−23.122	0.45089	0.65729	0.06178	−30.42	−101.35	0.0664
3	−23.179	0.45239	0.65795	0.06140	−30.50	−101.88	0.0985
4	−23.257	0.45445	0.65886	0.06088	−30.58	−102.41	0.1338
5	−23.340	0.45666	0.65983	0.06033	−30.61	−102.65	0.1679
6	−23.457	0.45977	0.66118	0.05957	−30.55	−102.15	0.2153
7	−23.499	0.46089	0.66167	0.05929	−30.47	−101.62	0.2332
8	−23.529	0.46168	0.66201	0.05910	−30.40	−101.09	0.2465
9	−23.556	0.46239	0.66232	0.05893	−30.31	−100.48	0.2589
10	−23.573	0.4628	0.6625	0.0588	−30.243	−100.0	0.2673
11	−23.700	0.4662	0.6640	0.0580	−23.112	−50.0	0.5497
12	−23.422	0.4588	0.6608	0.0598	−15.660	0.0	0.6974
13	−23.042	0.4487	0.6563	0.0623	−7.764	50.0	0.8197
14	−22.590	0.4368	0.6510	0.6550	0.6947	100.0	0.9316
15	−19.798	0.3641	0.6165	0.0894	45.629	300.0	1.3894
16	−12.032	0.1923	0.5068	0.2124	119.294	300.0	1.9024
17	−7.056	0.1169	0.4290	0.3494	76.506	100.0	1.3727
18	−6.036	0.1048	0.4128	0.3829	42.543	50.0	1.0091
19	−4.225	0.0860	0.3839	0.4453	−5.1060	0.0	0.0612

TABLE 4

Values of partial derivatives $\frac{\partial g}{\partial v}$ *and* $\frac{\partial g}{\partial m}$ *where* $g = \det(2\mathbf{J} \odot \mathbf{I}_4)$. *Data is presented for partials computed by automatic differentiation (AD), the adjoint formula of* §2.4 *(AF), forward differencing (FD), and central differencing (CD). Each entry associated with a difference formula is followed by the mantissa of the stepsize.*

Meth	v		m	
AD	−54.2506431900141592		−7728.29850144535158	
AF	$-54.25064319001\underline{5}$		$-7728.29850144535\underline{3}$	
CD	$-54.250643\underline{2}$	(−4)	$-7728.298501\underline{5}$	(−6)
FD	$-54.2506\underline{3}$	(−6)	$-7728.2985\underline{3}$	(−8)

curve. The curve of Hopf points exhibits a narrow turning point with respect to the stimulus current I near one end. Points 1 through 10 resolve the turning point of the Hopf curve.

We give a brief comparison of methods for computing the partial derivatives of augmenting functions using this example. For what follows, let $y = (v, m, n, h, v_k)$ denote the vector of phase space variables augmented with the parameter v_k and $g(\cdot)$ the function for Hopf detection. We consider here the augmenting function labeled (BP) in Table 2, given by $g(y; I) = det\,(2\mathbf{J} \odot \mathbf{I}_n)$. Recall that the standard Newton-type corrector step requires the solution of a linear system, the matrix of which contains partial derivatives of the form $\frac{\partial}{\partial y_i} g(y)$. As in most examples, we cannot determine exactly the values of the relevant partial derivatives at arbitrary points along the Hopf branch. Our objective here is to compare the common techniques for computing derivatives as applied to the Hodgkin–Huxley equation and comment on their distinguishing characteristics. The Hodgkin–Huxley equations are well suited to this comparison because expressions for the higher-order derivatives of the defining equations, although lengthy, may be easily formed. For example, values for the partial derivatives $\frac{\partial}{\partial v}\,(2\mathbf{J} \odot \mathbf{I}_4)$ may be computed using the adjoint formula given in [11]. Table 4 shows a compilation of these partial derivatives, computed for point 14 of Table 3. In our experiments, the adjoint matrix was formed in a general fashion, using determinants computed using LU factorization of appropriate submatrices.

Automatic differentiation provides an alternative to difference formulas for the calculation of derivative data. Neither symbolic algebra nor numerical differencing derivatives are computed essentially by algorithmic application of the chain rule to a reduced (optimized) representation of the operation sequence. Several implementations are available and the approach has been successfully tested on a variety of problems in engineering and operations research. Data presented in Table 4 was produced with the package ADOL-C [10] using double precision (53 significand bits, 15–17 decimal digits). The partial derivative values computed by automatic differentiation agree with those calculated using the adjoint formula to 14 decimal digits, as indicated by the underscore beneath the first differing digit. These results are especially interesting because no formula for second derivative data is required, merely a slightly modified version of the routine which evaluates the expression for $g(\cdot)$. In our tests, a 16.4 Kbyte temporary storage buffer was required by the ADOL-C package for the partial derivative calculation. This appears to be the primary disadvantage to automatic differentiation: the storage requirements for the derivative calculations can be quite substantial, even for relatively simple functions.

A general-purpose package for numerical pathfollowing of equilibria typically contains a facility for computing the augmented Jacobian using finite differencing. Here the usual caveats concerning numerical differentiation apply. Finite difference approximation is unreliable, especially applied to functions where accumulated roundoff error is large. High-order formulae require multiple function evaluations. The difference in stepsize must be chosen to properly balance the effects of roundoff and truncation error. Table 4 shows a compilation of the partial derivatives, $\frac{\partial}{\partial v}\det(2\mathbf{J} \odot \mathbf{I}_n)$ and $\frac{\partial}{\partial m}\det(2\mathbf{J} \odot \mathbf{I}_n)$, computed using both forward (FD) and central (CD) difference formulae for the selected Hopf point. Next to each entry (in parenthesis) is the exponent of the stepsize h which produces the numerical derivative closest to that computed by automatic differentiation. The computed error from stepsize choices substantially larger than h are due to truncation error; stepsizes significantly smaller exhibit roundoff error and, finally, catastrophic cancellation.

3.2. A model of a bursting neuron. The second example we study is a dynamical system used to describe the electrical activity of the anterior burster (AB) neuron in the stomatogastric ganglion of the lobster *Panulirus interruptus*. Based on work of Plant [19] and Rinzel and Lee [20], the model is similar in structure to the Hodgkin–Huxley equations but more complex. Under certain physiological conditions the AB cell produces complicated, rhythmic patterns of action potentials. Guckenheimer, Gueron, and Harris-Warrick [12] have shown that some of this behavior is well described by a six-dimensional system of ordinary differential equations, referred to in what follows as the Rinzel–Lee/A-current (RLA) model for the AB neuron. The model is obtained from that of Rinzel and Lee [20] by a change of time scale and the incorporation of an additional potassium ion current called the A-current.

Let $x = (x_1, \ldots, x_6) \in \mathbb{R}^6$ denote the vector of independent variables. Components of this vector correspond to the physical model as follows: x_1 is the voltage difference across the cell membrane; x_2 is a dimensionless quantity describing the activity of intracellular free calcium; x_3 controls the activation of the delayed-rectifier potassium channel; and x_4 controls the inactivation of the sodium channel; x_5 controls the activation of the calcium channel; and x_6 controls the inactivation of the A-current channel. It is convenient to define the model in terms of exponential functions that we denote by

$$\phi_{\pm}(x_1; \alpha, \beta) \overset{\text{def}}{=} \left(1 \pm e^{\alpha+\beta x_1}\right)^{-1}.$$

TABLE 5
Experimentally derived constants used in the AB cell model equations.

Subscript	α	β	γ	δ
1	−2.8714	−0.12095	−2.9841	−0.06196
2	–	–	−0.46154	−0.46154
3	−2.3714	−0.12095	−0.42143	−0.015119
4	−2.6857	−0.060476	−2.3714	−0.12095
5	−7.5	−0.15	10.333	0.16667

g_{Na}	$15.0\,\mu S$	v_{na}	$30.0\,mV$	g_{ca}	$0.04\,\mu S$
g_A	$8.0\,\mu S$	v_k	$-75.0\,mV$	g_l	$0.0854\,\mu S$
k_{ca}	$0.0078\,mV^{-1}$	v_{ca}	$140.0\,mV$	v_l	$-40.0\,mV$

The vector field describing the electrical state of the semipermeable membrane is then given by

$$\begin{aligned}
\dot{x}_1 &= -g_{Na}\varphi_2^3(x_1)\,x_4\,(x_1 - v_{na}) - 2g_{Ca}x_5\frac{(x_1 - v_{ca})}{(1+2x_2)} - g_K x_3^4\,(x_1 - v_k) \\
&\quad -2g_{KCa}x_2\frac{(x_1 - v_k)}{(1+2x_2)} - g_A\psi_2^3(x_1)\,x_6\,(x_1 - v_k) - g_l\,(x_1 - v_l)\,, \\
\dot{x}_2 &= -0.003\left[x_2 - k_{ca}x_5\frac{(x_1 - v_{ca})}{(1+2x_2)}\right], \\
\dot{x}_3 &= 0.8\left[(1 - x_3)\,\varphi_3(x_1) - x_3\psi_3(x_1)\right], \\
\dot{x}_4 &= 0.8\left[(1 - x_4)\,\varphi_4(x_1) - x_4\psi_4(x_1)\right], \\
\dot{x}_5 &= -.042553\left[x_1 - \phi_+(x_1;\alpha_5,\beta_5)\right], \\
\dot{x}_6 &= \phi_+(x_1;\gamma_5,\delta_5) - x_6
\end{aligned}$$

where

$$\begin{aligned}
\varphi_1(x_1) &= -(\alpha_1 + \beta_1 x_1)\,\phi_-(x_1;\alpha_1,\beta_1)\,, & \psi_1(x_1) &= 4e^{\gamma_1+\delta_1 x_1}, \\
\varphi_2(x_1) &= \frac{\varphi_1(x_1)}{(\varphi_1(x_1) + \psi_1(x_1))}, & \psi_2(x_1) &= \phi_+(x_1;\gamma_2,\delta_2)\,, \\
\varphi_3(x_1) &= -0.1\,(\alpha_3 + \beta_3 x_1)\,\phi_-(x_1;\alpha_3,\beta_3)\,, & \psi_3(x_1) &= 0.125e^{\gamma_3+\delta_3 x_1}, \\
\varphi_4(x_1) &= 0.07e^{\alpha_4+\beta_4 x_1}, & \psi_4(x_1) &= \phi_+(x_1;\gamma_4,\delta_4)\,.
\end{aligned}$$

The constants which enter these expressions through the functions $\phi_\pm$ were matched with the observed rates of activation and inactivation of ion channels in voltage-clamp experiments from other biological systems. Table 5 displays the values used in the numerical tests presented here. In addition, the model contains 11 physiological parameters which describe the various channel conductances and ion-reversal potentials. For our purposes, all parameters but (g_{KCa}, g_A) remain fixed at the nominal values shown in Table 5.

Numerical experiments for the RLA example were performed using a simple predictor-corrector algorithm implemented in the MATLAB© computation environment. Given an initial point at (or near) a solution, a sequence of points is advanced along the Hopf curve in a two-phase process: first, an Euler predictor step is taken using tangent data computed at

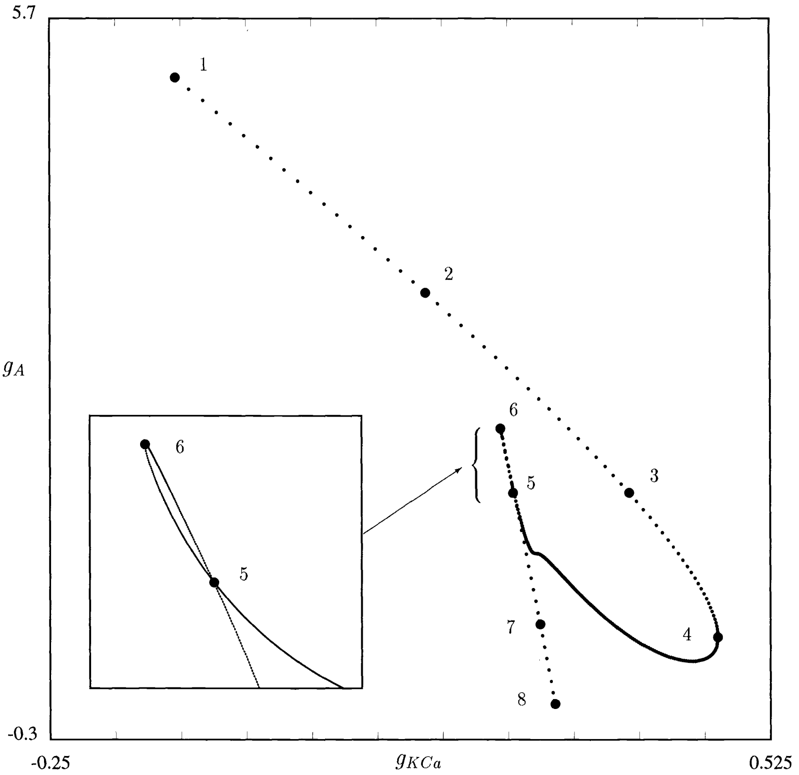


FIG. 1. *Curve of Hopf bifurcation points in the two-parameter plane (g_{KCa}, g_A) for the RLA neuron model computed using the Bezout resultant augmenting function (RB). Eight selected points corresponding to the entries in Table* 6 *are distinguished along the curve. The segment of the solution curve shown in the insert has been transformed by a simple rotation to make the closed loop visible at these scales.*

a previous solution. Then a series of Newton corrector steps is used to return the computed point on the bifurcation set. The length of the step taken between two successive solutions is determined by an algorithm of Georg [7], which seeks to maintain the initial rate of progress (contraction of the objective function values) exhibited by the sequence of corrector steps at a prescribed constant. The program design is very similar to that described by Allgower and Georg [1], suitably adapted for the current problem and MATLAB© implementation. To avoid the problems inherent in following Hopf curves which have turning points, we use pseudo-arclength continuation throughout. Using the notation introduced in (4), if $y = (x, g_{KCa}, g_A)$ is the vector of independent variables parameterized by τ, $y(\tau_i)$ is the ith Hopf solution point, $h = \tau_{i+1} - \tau_i$ is the desired steplength, and g is one of the defining equations shown in Table 2, then numerical continuation was performed using the augmented system

$$F(x, g_{KCa}, g_A; \tau) = \begin{pmatrix} f(x, g_{KCa}, g_A) \\ g(x, g_{KCa}, g_A) \\ v^t(\tau_i)\cdot[y(\tau)-y(\tau_i)]-h \end{pmatrix}$$

where $v(\tau_i)$ is the (fixed) unit tangent vector computed at the ith solution.

Figure 1 shows the Hopf bifurcation curve computed using the Bezout resultant $g = \det(\mathcal{B})$ as the augmenting equation. Eight points have been selected and enumerated for comparative purposes. The first labeled point is the initial condition used to start the continuation

TABLE 6
Phase-space coordinate data for each selected point shown in Figure 1.

Pt	x_1	x_2	x_3	x_4	x_5	x_6
1	−57.51974	0.25087	0.02357	0.93874	0.24452	0.32154
2	−56.95448	0.26247	0.02506	0.93262	0.26054	0.30134
3	−55.88832	0.28476	0.02810	0.91951	0.29251	0.26530
4	−53.51959	0.33524	0.03610	0.88184	0.37100	0.19570
5	−35.18183	0.57398	0.19477	0.23292	0.90227	0.01132
6	−34.47902	0.57615	0.20525	0.21286	0.91118	0.01008
7	−35.15696	0.57406	0.19513	0.23218	0.90260	0.01128
8	−34.95203	0.57473	0.19816	0.22620	0.90527	0.01090

TABLE 7
Parameter space coordinates (g_{KCa}, g_A) *for each selected point shown in Figure* 1. *Also shown are the magnitudes of the imaginary parts for the critical eigenvalues.*

Pt	g_{KCa}	g_A	$\lvert\mathrm{Re}(\lambda)\rvert$	$\lvert\mathrm{Im}(\lambda)\rvert$
1	−0.114740	5.207732	$<10^{-13}$	0.00125
2	0.154625	3.416376	$<10^{-13}$	0.00694
3	0.373380	1.753603	$<10^{-14}$	0.00942
4	0.468639	0.552728	$<10^{-14}$	0.01049
5	0.248306	1.797327	$<10^{-11}$ $<10^{-4}$	0.01241 0.06766
6	0.235546	2.269719	$<10^{-7}$	0.02252
7	0.270302	0.949501	$<10^{-11}$	0.10536
8	0.286695	2.805522	$<10^{-11}$	0.12937

algorithm, chosen very near the Takens–Bogdanov point which terminates the branch. Points 4 and 6 are turning points in the Hopf solution curve with respect to the parameter g_{KCa}, and point 5 marks a self-crossing of the Hopf branch that gives rise to a double Hopf bifurcation point. The insert provides a more detailed plot of the region near this degenerate point. Table 6 displays the phase-space coordinates of the selected points while Table 7 shows the parameter space coordinates of each selected point together with the magnitudes of the real and imaginary components of the critical eigenvalues.

The loop associated with the double point labeled 5 in the computed Hopf curve is a bit paradoxical. At the double point, there are two pairs of pure imaginary eigenvalues, and continuation around the loop must transform one of these pairs into the other. The mechanism for this transformation involves the geometry of how eigenvalues and eigenvectors depend upon matrices. The set of real $n \times n$ matrices with simple eigenvalues is not simply connected. When following a homotopically nontrivial loop in this set, eigenvalues of a matrix may be permuted. Such an interchange happens in this example. There is a point in the (g_{KCa}, g_A) plane inside the loop of the Hopf curve at which the system has a complex pair of double eigenvalues.

Our numerical experiments using the Bezout augmenting equation with the RLA model indicate that the numerical computation of the coefficients for the Jacobian characteristic polynomial $p(\lambda) = \sum_{k=0}^{5} c_k\lambda^k$, using reduction from the Hessenberg to Frobenius form by simple elimination [26], is reliable and accurate. To explore this observation further, we make the following ansatz: since the form of the RLA equations permits the explicit symbolic computation of the coefficients $\{c_k\}_0^5$ as functions of the Jacobian entries, we presume that these expressions, evaluated in extended floating point arithmetic at an equilibrium point,

TABLE 8

Condition numbers (κ_2) for the Jacobian ($\mathbf{J} = D_x f$), the corresponding companion matrix, and the spectral condition number s_λ associated with the pure imaginary eigenvalues.

Pt	s_λ	$\kappa_2(D_x f)$	$\kappa_2(C)$
1	1.0×10^{-2}	2.8×10^{6}	5.8×10^{8}
2	1.7×10^{-2}	2.5×10^{5}	1.7×10^{7}
3	1.5×10^{-2}	2.1×10^{5}	8.6×10^{6}
4	1.4×10^{-2}	1.9×10^{5}	6.8×10^{6}
5	2.0×10^{-3} 4.1×10^{-3}	6.4×10^{6}	2.4×10^{7}
6	9.0×10^{-4}	2.9×10^{7}	2.9×10^{7}
7	9.0×10^{-3}	1.6×10^{6}	9.7×10^{6}
8	1.2×10^{-2}	4.8×10^{5}	6.2×10^{6}

are exact. Moreover, we assume that the eigenvalues and eigenvectors computed using the unsymmetric QR algorithm are exact as well. Using these data we may evaluate how well conditioned the Hopf continuation task is both as an eigenvalue problem and, separately, as a polynomial rootfinding problem. The condition of the eigenproblem is equivalent to the sensitivity of the spectrum of the Jacobian to small perturbations of its elements ($\mathbf{J} + \epsilon\mathbf{E}$) reflected in the spectral condition number

$$s(\lambda) = \frac{y^H \cdot x}{||y||_2||x||_2}$$

where y and x are left and right eigenvectors of $\mathbf{J}$, respectively, of the eigenvalue λ. We expect that, for ϵ small, the perturbation of the eigenvalues will be less than $\delta\lambda = \epsilon||\mathbf{B}||_2/s(\lambda)$, and we consider the eigenproblem well conditioned if $s(\lambda)$ is near one. The first column of Table 8 presents the spectral condition number for the critical eigenvalues of the Jacobian at each selected point solution. Clearly, the data show that methods based on explicit determination of the spectrum or their sums directly from the Jacobian entries should be relatively insensitive to small perturbations to the elements of $\mathbf{J}$.

Table 8 also shows 2-norm condition estimates for the Jacobian matrix and its associated companion matrix as computed by Algorithm 1 at the selected points; in each case the reduction results in an inflation of the condition number, but the increase is mild (at most three orders of magnitude). Thus, the main source of instability in the computation of the characteristic polynomial coefficients—the use of nonunitary similarity transformations in the reduction of the Hessenberg form—is well behaved along the branch of Hopf points. We further examine the classical error estimates of the eigenvalues to the perturbation problem ($\mathcal{C} + \epsilon\mathbf{E}$) where $\mathcal{C}$ is the companion matrix associated with $\mathbf{J}$ and $\mathbf{E}$ is the matrix that contains a single nonzero entry in its nth column (which we take to be one). Now suppose λ_i is a simple root of the characteristic polynomial $charpoly(\mathcal{C}) = p(\lambda) = charpoly(\mathbf{J})$ for $1 \leq i \leq n$. A small perturbation in the kth coefficient of p

$$p(\lambda) + \epsilon\lambda^k = \lambda^6 + (c_k + \epsilon)\,\lambda^k + \sum_{i \neq k} c_i \lambda^i$$

may be related to the resulting perturbed root by the linear estimate

$$\lambda_i + \delta \approx \lambda_i - \epsilon \frac{\lambda_i^k}{\prod_{j \neq i} (\lambda_i - \lambda_j)}.$$

TABLE 9

Characteristic polynomial coefficients of maximal sensitivity for each selected point shown in Figure 1. Data is shown for each of three problems: $p(\lambda) = 0$, $r_e(z) = 0$, and $r_o(z) = 0$ where $z = \lambda^2$.

	$p(\lambda)$			$r_e(z)$			$r_o(z)$		
Pt	k	c_k	$s_k(\lambda)$	k	c_k	$s_k(\lambda)$	k	c_k	$s_k(\lambda)$
1	2	7.0e−3	8.9e−2	2	7.0e−3	2.2e−4	3	1.4e−1	2.5e−5
2	2	6.8e−3	5.1e−1	2	6.8e−3	7.1e−3	3	1.3e−1	7.3e−4
3	2	6.5e−3	7.2e−1	2	6.5e−3	1.4e−2	3	1.2e−1	1.3e−3
4	2	5.9e−3	9.0e−1	2	5.9e−3	1.9e−2	3	1.1e−1	1.6e−3
5	2	7.4e−4	8.9 43.7	2	7.4e−4	2.2e−1 6.8	3	6.0e−3	3.5e−2 1.0
6	2	2.2e−4	53.6	2	2.2e−4	4.2	1	−3.5e−6	163.6
7	1	3.8e−5	676.5	4	1.9e−1	6.9e−2	3	1.7e−2	1.3
8	4	1.9e−1	3.3e−1	4	1.9e−1	1.1e−1	3	1.2	2.1e−2

Thus, the relative sensitivity of each *simple* eigenvalue of **J** (equivalently, each simple root of p) to small perturbations in the coefficients of the characteristic polynomial are estimated by the magnitudes of the n factors s_k given by

$$s_k(\lambda_i) \triangleq \left| \frac{\lambda_i^k}{\prod\limits_{j \neq i} (\lambda_i - \lambda_j)} \right| .$$

The first three columns of Table 9 show, for each selected point, the subscript (k), value (c_k), and factor (s_k) for the coefficient of p upon which the critical eigenvalue is most sensitively dependent. For example, in the case of the last point (8), a perturbation of 1% in the value of c_4 will produce a change of approximately 0.5% in the computed critical eigenvalue. These data support the conclusion that relatively small errors in the numerically computed values of the characteristic polynomial coefficients produce only mild perturbations in the roots of p or, equivalently, the computed eigenvalues of **J**. Thus, construction of the residual polynomials in the first method of Kubiček (K1) may be expected to be well behaved.

From §2.1, the resultant methods are based on a specific rootfinding problem, that is, to find $z = \lambda^2$ which simultaneously satisfies

$$\left\{ \begin{array}{c} r_e(z) \\ r_o(z) \end{array} \right\} = \left\{ \begin{array}{c} z^3 + c_4 z^2 + c_2 z + c_0 \\ c_5 z^2 + c_3 z + c_1 \end{array} \right\} = 0.$$

Thus, sensitivity of the shared root to perturbations in the coefficients of the two separate polynomials is of interest, and the sensitivity estimates computed above for p may be extended as well to the polynomial pair (r_e, r_o) associated with p. The last six columns of Table 9 display, for each selected point, the values of s_k for the shared root corresponding to the coefficient of maximal sensitivity.

It is usual, in the discussion of Hopf pathfollowing using direct methods, to assume that a solution $(x^*, \alpha^*; \beta^*)$ on a branch is known and then to proceed to describe how subsequent solutions may be computed beginning with data from this point. However, practical experience suggests that frequently the task of finding one (or more) initial points is *by far* the most time-consuming part of a numerical bifurcation study, especially as the dimension of the phase and parameter spaces increases. We present a comparison of the rootfinding convergence regions for the augmented systems based on the Bezout resultant (labeled RB in Table 2) and the iterative eigensubspace method (JGR) for the RLA model.

The Bezout resultant is well defined at any point in the product space $\mathbb{R}^6 \times \mathbb{R}^2$, and all the information required to begin a corrector sequence at (x_0, g^0_{KCa}, g^0_A) may be derived from the linearization of the vector field at that point. This is not the case for the (JGR) system, which requires estimates for a basis $\{v, w\}$ of the eigenspace associated with the pure imaginary eigenvalue $\pm\omega i$ and a vector l not in $span\,\{v, w\}$. The intuitive choice is to use seed values v_0, w_0, ω_0, and l based on the complex elements of $spec(\mathbf{J})$ with the smallest real part. However, this eigenpair may not become critical for nearby values of (g_{KCa}, g_A) and an incorrect choice can bias the starting conditions away from the solution manifold, requiring many Newton steps to compute a solution, if convergence is achieved at all.

We explore this possibility in a small region of the (g_{KCa}, g_A) parameter space to the right of the double Hopf point in Figure 1. Two segments of Hopf bifurcation (referred to as the upper and lower branches in what follows) intersect at the double Hopf point. The RLA vector field has a single equilibrium for each choice of (g_{KCa}, g_A) in this region; moreover, the spectrum of the Jacobian for each such choice contains at least one complex eigenvalue pair. Since the rate at which the two pairs of complex eigenvalues cross the imaginary axis is very different, the intuitive choice is "correct" only in a thin band above the lower Hopf branch. If a Newton corrector is applied to choices for (g_{KCa}, g_A) in this parameter region, the results reflect the underlying structure in the spectrum of the equilibria, as shown in Figure 2. The lower plot shows data collected using the augmented system (JGR). A 75×75 point grid was established over the two-parameter regime. For each point in the grid, the unique equilibrium was found and the complete spectrum was computed. The complex eigenpair closest to the imaginary axis was used to compute starting values for the augmented variables and Newton's method was used to solve the nonlinear systems with $g(\cdot)$ chosen as in the (JGR) system of Table 2. The convergence criterion used was $||\mathbf{D}F||_\infty < TOL$, where $\mathbf{D}$ was a *constant* diagonal-scaling matrix chosen to equilibrate the disparate magnitudes of the various components in the objective function. If convergence was not achieved within 30 Newton steps, the trial was considered a failure. Successful trials are labeled with a triangle if they converged to the upper branch, a circle if to the lower branch. Thus, the plot shows the ultimate fate of a choice (g_{KCa}, g_A) given perfect information for the Jacobian at the starting conditions.

As expected, the set of converged initial conditions divides the parameter grid into roughly two parts: those which converge to the upper branch and those to the lower branch. Below the lower branch, the Jacobian has one complex eigenpair and computing the initial augmented variables based on the spectrum at the gridpoint is usually successful. In between, the dividing line occurs along the locus of intersection for the real parts of the two eigenpairs. Since this locus occurs very close to the lower branch, a disproportionate fraction of successful trials starting near the lower branch leave the neighborhood of the nearest Hopf bifurcations U. Instead of correcting the errant starting conditions for v_0, w_0, ω_0, the Newton sequence "corrects" the phase and parameter coordinates. In this example, computing v_0, w_0, ω_0 for starting conditions very near the lower branch using the "wrong" eigenpair still frequently results in a converged solution. In the general case, there is no reason why the delusive eigenpair need cross the imaginary axis at all; indeed, such a situation would typically be expected to result in convergence failure.

The analogous experiment was performed using the Bezout resultant $g = \det(\mathcal{B})$; the results are displayed in the upper plot of Figure 2. Again, the parameter region is divided by points which converge to the upper and lower Hopf branches. In this case, however, the distribution is more even. In particular, the lower branch exhibits a robust convergence neighborhood. A band of initial conditions which do not result in successful Newton sequences

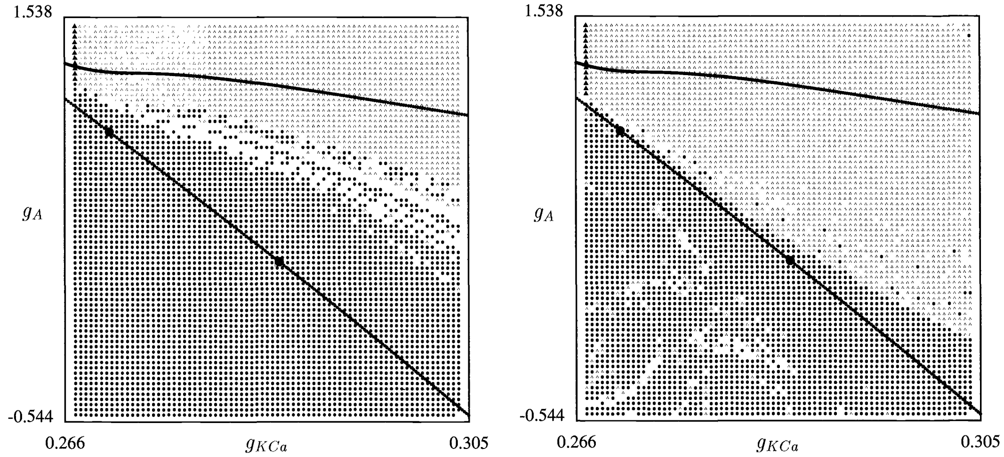


FIG. 2. *Convergence data for Newton's method in a neighborhood of Hopf branches in the* (g_{KCa}, g_A) *parameter plane. The left plot shows data for the Bezout resultant (RB) as the augmented function; the right plot shows data for the method of Jepson–Griewank–Reddien (JGR). Triangle symbols show initial conditions which ultimately converge to the upper Hopf branch while circles indicate convergence to a lower branch point.*

appears between the two Hopf branches, a zone outside the region of local convergence for Newton's method for any point in Γ. Such a failure zone is expected for any choice of augmenting functions and occurs where the quadratic model for the objective function F ceases to be sufficient to ensure adequate descent. A natural algorithmic improvement is to employ globally convergent variants of Newton's method to improve the behavior far from the solution set. In this context, a properly constructed quasi-Newton method must involve two steps: first, the full Newton step must be computed, and appropriate criteria for sufficient decrease in F evaluated. If the Newton step results in satisfactory progress, the full step is taken; otherwise, a modified step is computed. A common criterion is to guarantee that the result of the candidate step be a (sizable) fraction of the predicted improvement based on the gradient of F at the current point. More precisely, if y_k is the kth step of a corrector sequence and δy is the Newton step, then the algorithm accepts the full step if

$$F(y_k + \delta y) < F(y_k) + \epsilon[\nabla F(y_k)]^t \delta y \tag{15}$$

for some $0 < \epsilon < 1$. Typically, ϵ is chosen quite small, but it is significant that the obvious selection $\epsilon = 0$ is insufficient to ensure global convergence; some positive improvement in the objective function is required.

Figure 3 shows the result of applying the criterion (15) to the convergence experiment described above. At each step in the corrector sequence, the acceptance criterion was evaluated with $\epsilon = 0$; the point was deemed a failure if the inequality was not satisfied. Points which survive as successful trials not only converged to a Hopf bifurcation point but also made monotonic progress toward the solution at each corrector step. Thus, if a steplength adjustment (e.g., line search or dogleg) algorithm were used to improve global convergence behavior based on the acceptance criteria above (for $\alpha = 0$), the full Newton step would be accepted at each step. The results for the augmented function $g = \det(\mathcal{B})$ indicate that points near the border of the nonconvergence zone of Figure 2 are characterized by Newton steps which, early in the corrector sequence, do not strictly improve the objective function. However, apart from this band, most initial conditions are within the local convergence neighborhood. For the

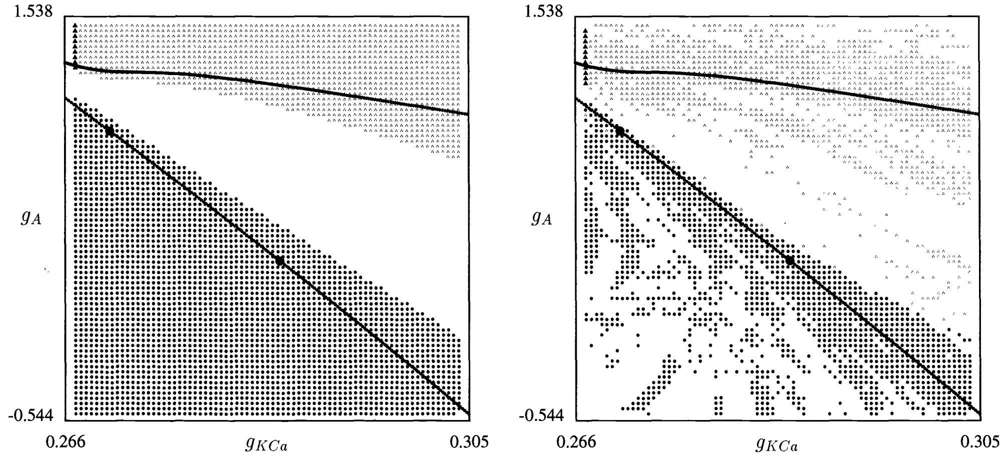


FIG. 3. *Convergence data for Newton's method in a neighborhood of Hopf branches in the* (g_{KCa}, g_A) *parameter plane including a monotonicity condition on the sequence of corrector steps. The left plot shows data for the Bezout resultant (RB) as the augmented function; the right plot shows data for the method of Jepson–Griewank–Reddien (JGR). Triangle symbols show initial conditions which ultimately converge to the left Hopf branch while circles indicate convergence to a right branch point.*

(JGR) augmenting function, nearly 50% of the trials which converge to a Hopf point without the monotonicity condition are *rejected* according to step acceptance criteria. This may be compared with a 23% reduction for the resultant method.

We regard the questions of accuracy and convergence considered above as more important than the speed of algorithms that detect Hopf detection in examples of moderate size. Nonetheless, we present some brief comparative data for our implementations of three algorithms within MATLAB©. For each of the Bezout resultant, biproduct matrix, and JGR methods from Table 2 we computed 20 points along a Hopf bifurcation curve shown in Figure 1 starting at a point along the curve and proceeding with an identical continuation algorithm. The floating point operation counts were comparable for the Bezout (664777 flops) and JGR methods (701244 flops) while the biproduct calculation was substantially slower (2786163 flops). Note, however, that the biproduct calculations did not exploit the sparsity of the biproduct matrix in computing its determinant. The biproduct method seems to offer the best opportunity to exploit the calculation of low-dimensional invariant subspaces in high-dimensional problems.

3.3. A larger neural model. Based on extensive experimental results, Golowasch [8] proposed a 14-dimensional system of ordinary differential equations as a model for the LP neuron in the stomatogastric ganglion of the crab *Cancer borealis* that incorporates the effects of eight separate ionic currents. The parameters which govern the activation kinetics for the associated conductance channels were derived from space-clamped experimental data. Buchholtz et al. [4] later amended this model; our third example is a 12-dimensional variation of these equations.

Despite the complexity of this system, analytic Jacobian derivatives were derived using Maple©. Using

$$\psi(u, v, \alpha) = e^{\frac{(u-v)}{\alpha}},$$
$$\phi(u, v, \alpha) = (1 + \psi(u, v, \alpha))^{-1}$$

the model equations for the LP neuron are given by

$$\begin{aligned}
\dot{x}_1 &= -c_{10}(g_{ca1}x_9x_{11} + g_{ca2}x_{10})(x_2 - \varphi_4(x_1)) + c_{20}(c_8 - x_1),\\
\dot{x}_2 &= (c_1 - ((g_{ca1}x_9x_{11} + g_{ca2}x_{10})(x_2 - \varphi_4(x_1)) + g_l(x_2 - e_l)\\
&\quad + g_d x_4^4(x_2 - e_k) + g_{KCa}x_5x_6(x_2 - e_k) + g_a x_8 x_7^3(x_2 - e_k)\\
&\quad + g_h x_{12}(x_2 - e_h) - c_{11}(\tfrac{1}{2}c_2(\varphi_5(x_2, x_3) - \varphi_6(x_2, x_3))))))/c_{11},\\
\dot{x}_3 &= k_h\left[c_6(1 - x_3)\psi(x_2, c_{44}, c_{27}) - x_3\phi(x_2, c_{50}, c_{33})\right],\\
\dot{x}_4 &= c_{12}\phi(x_2, c_{52}, c_{35})\,((\phi(x_2, c_{54}, c_{37}) - x_4),\\
\dot{x}_5 &= k_{oa}(\varphi_7(x_1, x_2) - x_5),\\
\dot{x}_6 &= c_{21}\left(\frac{c_4}{(c_5 + x_1)} - x_6\right),\\
\dot{x}_7 &= c_{16}(\phi(x_2, c_{41}, c_{24}) - x_7),\\
\dot{x}_8 &= c_{15}(\phi(x_2, c_{48}, c_{31}) - x_8),\\
\dot{x}_9 &= c_{17}(\phi(x_2, c_{42}, c_{25}) - x_9),\\
\dot{x}_{10} &= c_{18}(\phi(x_2, c_{43}, c_{26}) - x_{10}),\\
\dot{x}_{11} &= c_{19}(\phi(x_2, c_{49}, c_{32}) - x_{11}),\\
\dot{x}_{12} &= \frac{c_{13}(\phi(x_2, v_r, c_{38}) - x_{12})}{\phi(x_2, c_{53}, c_{36})},\\
\varphi_1(x_2) &= \frac{c_7(x_2 - c_{45})}{1 - \psi(x_2, c_{45}, c_{28})},\\
\varphi_2(x_2) &= \frac{\varphi_1(x_2)}{\varphi_1(x_1) + c_8\psi(x_1, c_{50}, c_{33})},\\
\varphi_3(x_1) &= \log\left[\psi(x_1, \tfrac{1}{100}, \tfrac{3}{100}) + \psi(x_1, -\tfrac{1}{100}, -\tfrac{3}{100})\right],\\
\varphi_4(x_1) &= c_{56} - c_{57}\ \log\left[\tfrac{2}{300}\varphi_4(x_1) + \tfrac{1}{3}(100x_1 - 1) + \tfrac{1}{100}\right],\\
\varphi_5(x_2, x_3) &= \frac{1}{c_2}\left(\frac{g_{Na}x_3\varphi_2^3(x_2)(e_{na} - x_2)}{c_{11}} - c_1\right),\\
\varphi_6(x_2, x_3) &= \log\left(e^{\varphi_5(x_2, x_3)} + e^{-\varphi_5(x_2, x_3)}\right),\\
\varphi_7(x_1, x_2) &= \frac{x_1}{(c_3 + x_1)}\phi(x_2, c_{46} - c_{14}x_1, c_{29})\phi(x_2, c_{47} - c_{14}x_1, c_{30}).
\end{aligned}$$

Table 10 lists the nominal values and units of the parameters.

The LP equations illustrate a difficulty that arises in using the determinant as an indicator of matrix singularity for methods based on polynomial resultants or biproducts. Extreme variations in the magnitude of det($\mathcal{B}$) on det($2\mathbf{I} \odot \mathbf{J}$) can adversely affect the behavior of numerical continuation methods, especially in large application problems. Moreover, since det($\cdot$) is not a true matrix condition indicator, poor scaling behavior in its (implicit) use as such can result in convergence failures. These remarks pertain not only to the use of det($\mathcal{B}$) or $2\mathbf{J} \odot \mathbf{I}$ in Hopf pathfollowing but also other applications which use the determinant as an objective function in rootfinding or optimization (detection of saddle-node bifurcations, for example). Thus, progress in addressing these issues in the context of Hopf continuation has implications to other applications, and vice versa.

The bifurcation diagram of the LP equations in the (I_{ext}, g_{Na}) plane has a fold in the surface of equilibrium points that produces a curve of saddle-node bifurcations. A cusp

TABLE 10
Values for the experimentally determined constants appearing in the LP model equations.

c_1	60000	c_{16}	142.857	c_{31}	6	c_{46}	0	e_l	−50
c_2	6000	c_{17}	50	c_{32}	8	c_{47}	16	g_d	0.35
c_3	2.5	c_{18}	10	c_{33}	−5	c_{48}	−70	g_{KCa}	3.2
c_4	0.7	c_{19}	16	c_{34}	−13	c_{49}	−50	g_a	1.7
c_5	0.65	c_{20}	360	c_{35}	−22	c_{50}	−40	g_{Ca1}	0.21
c_6	0.08	c_{21}	35	c_{36}	−13	c_{51}	−34	g_{Ca2}	0.047
c_7	0.11	c_{22}	166.6	c_{37}	−17	c_{52}	10	g_h	0.037
c_8	0.05	c_{23}	−12	c_{38}	7	c_{53}	−110	g_{Na}	2300
c_9	15	c_{24}	−26	c_{39}	15	c_{54}	−25	g_l	0.1
c_{10}	300	c_{25}	−7	c_{40}	−40	c_{55}	−7	k_{oa}	1
c_{11}	0.0017	c_{26}	−7	c_{41}	−12	c_{56}	115.47	k_h	600
c_{12}	180	c_{27}	−8	c_{42}	−11	c_{57}	12.19	v_r	500
c_{13}	0.33	c_{28}	−20	c_{43}	22	I	0	e_k	−80
c_{14}	0.6	c_{29}	−23	c_{44}	−39	e_h	−10		
c_{15}	20	c_{30}	−5	c_{45}	−6	e_{na}	50		

TABLE 11
Phase-space and parameter coordinates for three selected points along the Hopf curve for the LP equation. Also shown is the complete spectrum for each selected point.

	1	2	3	4
I_{ext}	−0.2099885	0.089912	5.557706	5.496061
g_{Na}	2268.22	1678.89	525.271	3347.40
x_1	0.146623	0.162821	0.196959	0.400147
x_2	−42.979	−39.121	−13.076429	3.657527
x_3	0.270205	0.129938	0.003136	0.000387
x_4	0.257771	0.303509	0.668495	0.843663
x_5	$3.41\cdot10^{-5}$	$9.383\cdot10^{-5}$	0.017156	0.073446
x_6	0.878709	0.861198	0.826486	0.666574
x_7	0.232991	0.260550	0.489651	0.646162
x_8	0.010949	0.005786	$7.581\cdot10^{-5}$	$4.66\cdot10^{-6}$
x_9	0.010268	0.017684	0.426381	0.890031
x_{10}	$9.30\cdot10^{-5}$	0.000161	0.006621	0.067840
x_{11}	0.293669	0.204259	0.009801	0.001221
x_{12}	$2.90\cdot10^{-4}$	0.000167	$4.05\cdot10^{-6}$	$3.71\cdot10^{-7}$
λ_1	−599.58	−598.88	−569.96	$10^{-8}+936.86i$
λ_2	−375.99	−377.33	−369.14	10^{-8}-$936.86i$
λ_3	−152.47	−203.60	$10^{-9}+278.76i$	−586.36
λ_4	−137.87	−141.86	$10^{-9}+278.76i$	−383.02
λ_5	−62.76	−60.72	−143.02	−142.86
λ_6	−35.09	−35.23	−131.49	−90.53
λ_7	−20.26	−20.10	−48.64	−50.28
λ_8	−15.47	−16.57	−32.69	−36.86
λ_9	−9.95	−9.95	−20.00	−20.00
λ_{10}	−2.09	$10^{-5}+8.99i$	−13.73	−15.93
λ_{11}	$10^{-4}+0.01i$	$10^{-5}-8.99i$	−11.01	−11.13
λ_{12}	$10^{-4}+0.01i$	−0.33	−0.33	−0.33

occurs along the fold, dividing the saddle-node bifurcation set into upper- and lower-branch segments. Along the upper branch, one of the fixed points annihilated in the saddle-node interaction undergoes a change of stability, producing a curve of Hopf points proximal to the saddle-node curve. The Hopf branch arises at a Takens–Bogdanov point and leaves the parameter regime of physiological interest.

Table 11 shows parameter coordinates for four selected points along the Hopf branch described above. Also listed are values for the phase-space variables and the complete spectrum. Notice that the critical eigenvalues begin with zero magnitude but become the eigenvalues of largest magnitude at point 4 along the branch of Hopf bifurcations and that there are several additional eigenvalues of large magnitude.

One difficulty is immediately apparent from these data: if $\lambda_i = \bar{\lambda}_{i+1}$ is the critical eigenpair and the remaining real eigenvalues are arranged in descending order, $\lambda_j \geq \lambda_{j+1}$ for $j \neq i, i+1$, then to achieve a prescribed solution tolerance $||F(x^*, I^*_{ext}, g^*_{Na})||_\infty < \epsilon$ requires

$$Re(\lambda_i) < \frac{\epsilon}{2}\left[\prod_{\substack{k=3\\k\neq i,i+1}}^{12} \lambda_k^{k-3}\left(\lambda_k^2 + |\lambda_i|\right)\right]^{-1}$$

using the rough estimate that $(\lambda_i + \lambda_j) \approx \max(\lambda_i, \lambda_j)$ for eigenvalue pairs that do not approach zero in a neighborhood of a solution. For large systems, or even small systems with a few large eigenvalues, this explosion in the magnitude of the biproduct determinant or resultant severely restricts the size of the neighborhood about the solution manifold where the computed value contains any significant digits. In the LP equations, the effect is prohibitive.

To apply the algebraic Hopf methods previously described to problems of this type requires a strategy for eliminating the disabling effects of the unwanted large eigenvalues, so long as they do not participate in the bifurcation. One possible approach involves substituting an alternative method for determining when $2\mathbf{J} \odot \mathbf{I}_n$ or $\det(\mathcal{B})$ is singular. For example, if a true condition estimate was substituted for $\det(\cdot)$, such as

$$g(x, I_{ext}, g_{Na}) = ||\,(2\mathbf{J} \odot \mathbf{I}_n)^{-1}\,||_F^{-1} \tag{16}$$

where $||\cdot||_F$ is the Frobenius norm, the required differentiability properties are retained. Another alternative, suggested by Allgower, Georg, and Miranda [2] in the context of using multidimensional resultants to compute real polynomial roots, is the function

$$g(x, I_{ext}, g_{Na}) = \min_{||u||=1} ||\,[2\mathbf{J} \odot \mathbf{I}_n]\,u||_2^2. \tag{17}$$

Notice that this is equivalent to iteratively driving the smallest singular value σ_n to zero and, thus, uses the square of the 2-norm condition number as the augmenting equation. This choice is especially attractive since σ_n is widely considered the most reliable estimate of matrix condition, and several different methods for its computation might be adapted to the problem of Hopf pathfollowing. In the continuation framework it may also be possible to make use of spectral information computed at wide intervals and updated cheaply at intermediate solution points. For example, suppose that at (or near) a solution $(x^0, I^0_{ext}, g^0_{Na})$, the full set of eigenvalues $\lambda_1, \ldots, \lambda_n$ and corresponding eigenvectors $v_1, \ldots, v_n$ are computed. Then, by the elementary properties of the biproduct matrix, a vector in the nullspace of $\mathbf{J} \odot \mathbf{I}_n$ can be formed: using the lexicographic ordering scheme introduced in §2.3, if v^i_p is used to denote the ith component of the pth eigenvector, we can construct an eigenvector V corresponding to the eigenvalue $\lambda_i + \lambda_j$ of $2\mathbf{J} \odot I_n$ according to the formula

$$V^{\{pq\}}_{\{ij\}} = \det\begin{bmatrix} v^p_i & v^p_j \\ v^q_i & v^q_j \end{bmatrix} \tag{18}$$

where $1 \leq j, p \leq n-1$ and $2 \leq i, q \leq n$. This vector may be stored and used to start a conjugate gradient algorithm (for example), constrained to the unit sphere, as an estimate for

u in (17) to compute subsequent corrector steps. (See [2] for a discussion of the conjugate gradient method applicable to this situation.)

An important special case arises when some of the large eigenvalues are associated with an invariant subspace which evolves slowly as the parameters are varied [25]. More precisely, suppose at a point $(x^0, I^0_{ext}, g^0_{Na})$ near a Hopf bifurcation the range of the (nonsingular) Jacobian matrix $\mathbf{J}_0$ may be split into a direct sum of eigenspaces $W_c \oplus W_h$ with $\dim(W_c) = m$ and $\dim(W_h) = n - m$ such that the following hold:

1. locally, W_h contains the two-dimensional subspace corresponding to the Hopf bifurcation;
2. W_c contains some of the large eigenvalues in the spectrum of $\mathbf{J}_0$; and
3. W_c remains nearly invariant as $\mathbf{J}$ is perturbed away from $\mathbf{J}_0$.

Then we may eliminate the unwanted effects of W_c altogether by deflating the Jacobian to form a new matrix that has only the eigenvalues associated with W_h. The literature on matrix deflation in the context of eigenvalue determination is vast; explicit techniques are central to iterative methods and implicit use of deflation is integral to various Lanczos schemes. In our preliminary experiments we used a simple algorithm based on similarity transformations. Suppose an $n \times m$ rectangular matrix $\mathbf{U}$ can be obtained with linearly independent columns so that $range(\mathbf{U}) = W_c$. Then applying Gaussian elimination we form the product $\mathbf{\Sigma}$ of elementary matrices that triangularize $\mathbf{U}$:

$$\mathbf{\Sigma U} \equiv \mathbf{QPU} = \left[\begin{array}{c} \mathbf{T} \\ --- \\ \mathbf{0} \end{array} \right]$$

where $\mathbf{T}$ is upper triangular and reference to the permutation matrix $\mathbf{P}$ is made explicit to emphasize the use of partial pivoting to stabilize the transformation. Applying $\mathbf{QP}$ to $\mathbf{J}$ produces a block-structured equation involving the Jacobian:

$$\mathbf{\Sigma J} = \left[\begin{array}{cc} \mathbf{J}_{11} & \mathbf{J}_{12} \\ \mathbf{E} & \mathbf{J}_{22} \end{array} \right] \mathbf{\Sigma}$$

where $\mathbf{J}_{11}$ is an $m \times m$ matrix. It follows easily that if $\mathbf{U}$ is chosen as above, $\mathbf{U}$ commutes with $\mathbf{J}$ if and only if $\mathbf{E} = \mathbf{0}$ and, if so, $spec\ (\mathbf{J}_{22})$ displays the same set of eigenvalues as does $\mathbf{J}$ on W_h (see Wilkinson [27, Chap. 9, §§21–24] for a discussion).

Given a candidate splitting $W_c \oplus W_h$, the sensitivity of the deflating subspace $range(\mathbf{U})$ to perturbations in the Jacobian matrix is properly expressed in terms of the *separation* of the matrices $\mathbf{J}_{11}$ and $\mathbf{J}_{22}$, measured by

$$sep\ (\mathbf{J}_{11}, \mathbf{J}_{22}) = \min_{\mathbf{X} \neq \mathbf{0}} \frac{||\mathbf{J}_{11}\mathbf{X} - \mathbf{X}\mathbf{J}_{22}||}{||\mathbf{X}||}$$

where $\mathbf{X}$ ranges over the subspace of $p \times (n - p)$ matrices. The $sep(\cdot, \cdot)$ function is difficult to compute, in general, but bounds exist based on the spectra of the matrix arguments (when $\mathbf{J}$ is diagonalizable):

$$\begin{aligned} s_l\ (\mathbf{J}_{11}, \mathbf{J}_{22}) &= \frac{\min |spec(\mathbf{J}_{11}) - spec(\mathbf{J}_{22})|}{\kappa_2\ (\mathbf{Q}_1)\, \kappa_2\ (\mathbf{Q}_2)}, \\ &\leq sep\ (\mathbf{J}_{11}, \mathbf{J}_{22}), \\ &\leq \min |spec(\mathbf{J}_{11}) - spec(\mathbf{J}_{22})| = s_u\ (\mathbf{J}_{11}, \mathbf{J}_{22}) \end{aligned}$$

TABLE 12
Separation function values and bounds for deflations of dimension two to seven for the LP model equations. Each p-dimensional deflating subspace contains the p-largest real eigenvalues of the Jacobian matrix at the initial branch point.

p	$s_u()$	$sep()$	$s_l()$
2	174	36	2
3	62	3	$6 \cdot 10^{-2}$
4	81	7.0	$8 \cdot 10^{-2}$
5	26	2.5	$2 \cdot 10^{-2}$
6	15	$4 \cdot 10^{-1}$	$2 \cdot 10^{-3}$
7	4	$3 \cdot 10^{-1}$	$4 \cdot 10^{-4}$

where $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are chosen to diagonalize $\mathbf{J}_{11}$ and $\mathbf{J}_{22}$, respectively. Equality holds on the right when $\mathbf{J}$ is normal, but, in general, the separation of $\mathbf{J}_{11}$ and $\mathbf{J}_{22}$ may be much less than the minimum distance between the respective spectra. Procedures for the numerical estimation of $sep(\cdot, \cdot)$ have been proposed [5], similar in form to well-known methods of matrix condition estimation. Table 12 shows values for $s_l(\cdot, \cdot)$, $sep(\cdot, \cdot)$, and $s_u(\cdot, \cdot)$ for six choices of W_c of increasing dimension m for the LP model Jacobian. Each case corresponds to a choice for W_c that contains the m-largest real eigenvalues of $\mathbf{J}$ evaluated at the second point of Table 11.

A simple strategy for incorporating deflation of the Jacobian into numerical Hopf pathfollowing is the following: suppose the complete eigenstructure for $\mathbf{J}$ is computed at the initial solution point on the Hopf branch (by any method), m is determined by $sep()$ estimates, and the columns of $\mathbf{U}$ are chosen to be the appropriate eigenvectors of $\mathbf{J}$. At each subsequent corrector step $||\mathbf{E}||_F$ is computed and, if sufficiently large, the columns of $\mathbf{U}$ are updated using a small number of inverse iteration steps. Notice that each column may be adjusted independently, so if $m >> (n - m)$ and n is of moderate size, this step can be performed in parallel. Numerical continuation was begun at selected point 2 of Table 11 using g_{Na} as the continuation parameter and proceeded toward the Takens–Bogdanov point at the beginning of the branch (near point 1). The seven-dimensional deflating subspace of Table 12 was used and updated at each corrector step. The residual real part of the critical eigenpair is below 10^{-12} near the beginning of the curve and remains below 10^{-9} as the solutions approach the Takens–Bogdanov bifurcation.

4. Concluding remarks. We have examined the application of minimal augmentation methods for computing Hopf bifurcations to three examples of vector fields that describe the electrical activity of axons and neurons. There are several conclusions we draw from this work.

1. The derivation of symbolic expressions for the detection of Hopf bifurcations in families of vector fields of moderate complexity is feasible. The usefulness of these symbolic expressions is dependent upon the complexity of expressions for the Jacobian of a vector field and upon the effectiveness of rootfinding algorithms.

2. Automatic differentiation algorithms work. They give far more accurate values for derivatives than are obtainable with the simplest finite difference methods of computing derivatives.

3. Continuation applied to minimal augmentation methods for finding Hopf bifurcations appears to work at the singularities associated with double Hopf bifurcations in two-parameter families of vector fields. The rootfinding problem has a singularity of corank 1 at these points, but the method appears to follow the smooth branches of solutions passing through this point with little difficulty. This success calls for further theoretical justification.

4. In at least some circumstances, the minimal augmentation methods are preferable to direct methods that require the computation of eigenvectors. In the example we present, iterative rootfinding algorithms that start with natural choices for their initial seeds converge more reliably and have larger regions of convergence for the minimal augmentation methods than for the direct methods that solve for eigenvectors.

5. In problems that are large or stiff, straightforward implementation of the minimal augmentation methods lead to rootfinding problems that are poorly conditioned. The difficulty lies in the determination of when a large matrix with eigenvalues of large magnitude is singular. By using the extensive numerical linear algebra theories for calculating the condition number of a matrix, the algorithms for detecting Hopf bifurcations can be significantly improved. Implementations based on calculations of the smallest singular value of a matrix appear to be an attractive target for further work in this area.

6. Despite the growth in the size of the linear algebra problems that are associated with the definition of the biproduct matrix as a strategy for computing Hopf bifurcations, the impact of this growth can be ameliorated by the intelligent use of standard algorithms for linear algebra problems. These increase the sparsity of the biproducts whose condition number must be calculated or localize these calculations to smaller matrices calculated from the biproduct.

Acknowledgments. Many of the calculations were performed with the software package DsTool [3] developed at Cornell University.

REFERENCES

[1] E. L. Allgower and K. Georg, *Numerical Continuation Methods*, Springer-Verlag, New York, 1990, pp. 267–287.

[2] E. L. Allgower, K. Georg, and R. Miranda, *The method of resultants for computing real solutions of polynomial systems*, SIAM J. Numer. Anal., 29 (1992), pp. 831–844.

[3] A. Back, G. Guckenheimer, M. Myers, F. Wicklin, and P. Worfolk, *dstool: Computer assisted exploration of dynamical systems*, AMS Notices, April, 1992, pp. 303–309.

[4] F. Buchholtz, J. Golowash, I. R. Epstein, and E. Marder, *Mathematical model of an identified stomatogastric ganglion neuron*, J. Neurophysiology, 67 (1992), pp. 332–340.

[5] R. Byers, *A LINPACK-Style condition estimator for the equation $AX - XB^t = C$*, IEEE Trans. Automat. Control, AC-29 (1984), pp. 926–928.

[6] R. Fitzhugh, *Impulses and physiological states in theoretical models of nerve membrane*, Biophysical J., 1 (1961), pp. 445–466.

[7] K. Georg, *A note on stepsize control for numerical curve following*, in Homotopy Methods and Global Convergence, NATO Conf. Series II, Systems Sci., 13, Plenum Press, New York, 1983, pp. 145–154.

[8] J. Golowasch and E. Marder, *Ionic currents of the lateral pyloric neuron of the stomatogastric ganglion of the crab*, J. Neurophysiology, 67 (1992), pp. 318–331.

[9] A. Griewank and G. Reddien, *The calculation of Hopf points by a direct method*, IMA J. Numer. Anal., 3 (1983), pp. 295–303.

[10] A. Griewank, D. Juedes, J. Srinivasan, and C. Tyner, *ADOL-C: A Package for the Automatic Differentiation of Algorithms Written in C/C++*, Argonne Tech. Report ANL-MCS-P180-1190, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, 1990.

[11] J. G. Guckenheimer, M. R. Myers, and B. Sturmfels, *Computing Hopf Bifurcations* I, SIAM J. Numer. Anal., 34 (1997), to appear.

[12] J. Guckenheimer, S. Gueron, and R. M. Harris-Warrick, *Mapping the dynamics of a conditional bursting neuron*, Philos. Trans. Roy. Soc. London, 341 (1993), pp. 345–359.

[13] M. Holodniok and M. Kubiaček, *New algorithms for the evaluation of complex bifurcation points in ordinary differential equations. A comparative numerical study*, Appl. Math. Comput., 15 (1984), pp. 261–274.

[14] A. L. Hodgkin and A. F. Huxley, *A quantitative description of membrane current and its applications to conduction and excitation in nerve*, J. Physiology, 117 (1952), pp. 500–544.

[15] A. I. Khibnik, *LINLBF: A program for the continuation and bifurcation analysis of equilieria up to codimension three*, in Continuation and Bifurcation: Numerical Techniques and Applications, D. Roose, ed., Kluwer Academic Publishers, Norwell, MA, 1990, pp. 283–296.

[16] M. Kubiček, *Algorithm for evaluation of complex bifurcation points in ordinary differential equations*, SIAM J. Appl. Math., 38 (1984), pp. 103–107.

[17] M. KUBIČEK, *Occurrence of oscillatory regimes in lumped parameter systems, determination of Hopf bifurcation points*, Chem. Engrg. Sci., 35 (1979), pp. 1078–1080.

[18] G. MOORE, T. J. GARRETT, AND A. SPENCE, *The numerical detection of Hopf bifurcation points*, in Continuation and Bifurcation: Numerical Techniques and Applications, D. Roose, ed., Kluwer Academic Publishers, Norwell, MA, 1990, pp. 227–246.

[19] R. E. PLANT, *Bifurcation and resonance in a model for bursting neurons*, J. Math. Bio., 11 (1981), pp. 15–32.

[20] J. RINZEL AND Y. S. LEE, *Dissection of a model for neuronal parabolic bursting*, J. Math. Bio., 25 (1987), pp. 653–675.

[21] J. RINZEL AND R. N. MILLER, *Numerical calculation of stable and unstable periodic solutions to the Hodgkin-Huxley equations*, Math. Biosci., 49 (1980), pp. 27–59.

[22] D. ROOSE AND V. HLAVACEK, *A direct Method for the computation of Hopf bifurcation points*, SIAM J. Appl. Math., 45 (1985), pp. 879–894.

[23] ———, *Numerical computation of Hopf bifurcation points for parabolic diffusion-reaction differential equations*, SIAM J. Appl. Math., 43, (1983), pp. 1075–1085.

[24] D. ROOSE, *An algorithm for the computation of Hopf bifurcation points in comparison with other methods*, J. Comput. Appl. Math., 12/13 (1985), pp. 517–529.

[25] G. W. STEWART, *Error and perturbation bounds for subspaces associated with certain eigenvalue problems*, SIAM Rev., 15 (1973), pp. 727–764.

[26] J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Monographs Numer. Anal., Oxford Science Publications, Oxford University Press, New York, 1965, pp. 405–408.

[27] ———, *The Algebraic Eigenvalue Problem*, Monographs Numer. Anal., Oxford Science Publications, Oxford University Press, New York, 1965, pp. 584–589.

SIAM J. SCI. COMPUT.
Vol. 17, No. 6, pp. 1302–1327, November 1996

EVOLUTION OF CONVEX PLANE CURVES DESCRIBING ANISOTROPIC MOTIONS OF PHASE INTERFACES*

KAROL MIKULA† AND JOZEF KAČUR‡

Abstract. The numerical approximation schemes for solving the nonlinear initial value problem

$$\partial_t b(v) = (Av_x)_x + (Bv)_x + Dv + G$$

with periodic boundary conditions are presented. We assume that the function b is increasing, and asymptotically $b'(s) = 0$ and $b'(s) = +\infty$, so that the model describes in a sense both slow and fast diffusions. The solution also may blow up in a finite time. This problem arises from the evolving curves theory, which was used in the construction of models of motion of phase interface in multiphase thermomechanics by Angenent and Gurtin [*Arch. Rational Mech. Anal.*, 108 (1989), pp. 323–391]. The so-called "curve shortening equation" is included in the model. Our approximating solutions converge strongly in $L_2(I, V)$ space to the weak solution. We also derive an "error estimate" for semidiscretization, which implies uniqueness. The numerical experiments in various situations of the "anisotropic curve shortening" are discussed.

Key words. curve shortening flow, phase interface, nonlinear degenerate parabolic equations, blow up, numerical solution, L_∞-estimate, error estimate

AMS subject classifications. 35K65, 65N40, 53C80

1. Introduction and the background of the problem.

The aim of this paper is to study from a numerical point of view the following problem. Let v be a smooth function which satisfies

$$\begin{aligned} \partial_t b(v) &= (Av_x)_x + (Bv)_x + Dv + G, \\ v(x,t) &= v(x + 2\pi\nu, t), \qquad (1.1) \\ v(x,0) &= v_0(x) \end{aligned}$$

for $x \in \mathbb{R}$ and $t \in [0, T)$, where $\nu \in \mathbb{N}$, v_0 is a smooth function, and

(H1) $A \geq q > 0$, B, B_x, D, G are bounded measurable functions of independent variable x, periodic in interval $[0, 2\pi\nu]$;

(H2) $b(s)$ is a C^2-function, defined in interval $(-\infty, a)$, $a < \infty$, satisfying $b'(s) > 0$,

$$\begin{aligned} &b(s) \to a_1 \text{ and } b'(s) \to 0 \text{ for } s \to -\infty, \\ &b(s) \to \infty \text{ and } b'(s) \to \infty \text{ for } s \to a^-, \end{aligned} \qquad \text{(H3)}$$

$$b(0) = 0. \qquad \text{(H4)}$$

So, the nonlinearity in this model is represented by the increasing function $b(s)$ admitting asymptotical *degeneracy of two types* expressed by hypothesis (H3). In this sense our model covers locally both slow and fast diffusions. Due to the special form of the problem, the solution also may *blow up* in a finite time. From a numerical point of view, complications also arise because of the strong *convection term* in the governed equation.

We suggest two approximation schemes for solving degenerate parabolic problems of the type (1.1). We numerically solve certain regularization of (1.1), which is identical with the original problem in a local time interval. Although our results are theoretically guaranteed only locally, numerical calculations show the agreement with known exact results in time intervals

*Received by the editors January 19, 1994; accepted for publication (in revised form) May 26, 1995.

†Department of Mathematics, Slovak Technical University, Radlinskeho 11, 813 68 Bratislava, Slovakia (mikula@cvtstu.cvt.stuba.sk).

‡Department of Numerical Analysis and Optimization, Faculty of Math and Physics, Comenius University, 842 15 Bratislava, Slovakia (kacur@fmph.uniba.sk).

with right endpoints very near to the blow-up time. The first of the schemes is simpler; we use it, besides other things, to establish the relation between original and regularized problems. The second scheme is more sophisticated; it is based on semidiscretization which reduces (1.1) to a succession of linear elliptic equations coupled with algebraic corrections due to nonlinearity. It originates in the works of Magenes, Nochetto, and Verdi [MNV] and Jäger and Kačur [JK1], [JK2] concerning the porous medium-type equations and elliptic–parabolic problems. For this scheme we not only prove the convergence but also derive an error estimate, which implies the uniqueness of the solution. In numerical computation these schemes yield results of comparable degrees of accuracy while the second one is more precise near the degeneracy $b'(s) = +\infty$. After time semidiscretization, both schemes need to solve a convection–diffusion equation in each time step. For this purpose we use the so called "power-law scheme" described in [P], which respects the "up-wind" principle.

Problem (1.1) arises from the curve evolution model describing anisotropic motion of phase interfaces in multiphase thermomechanics given by Angenent and Gurtin in [AG1]. Assume that at time t a solid occupies a region $D(t) \subset \mathbb{R}^2$ whose boundary is a smooth curve $\mathbf{r}$. Let θ be variable representing the angle of the tangent to the curve with x-axis. By Ω we denote the range of θ. Let β, g, F be functions on Ω coming from the constitutive description of interface and representing the following physical quantities. $\beta(\theta)$ is the kinetic coefficient; from thermodynamical considerations it follows that $\beta(\theta) > 0$. The function $g(\theta)$ is given by $g(\theta) = f(\theta) + f''(\theta)$, where $f(\theta)$ is the interfacial free energy. $F \equiv$ const is the energy of the solid phase minus that of the other phase (see [AG1], [AG2], [G1], [G2], [G3]). By [AG1], the first equation of the interface motion may be written as

$$\beta(\theta)\mathcal{V} = g(\theta)k - F, \tag{1.2}$$

where $\mathcal{V} \equiv \mathcal{V}(Q, t)$ is the normal velocity and $k(Q, t)$ the curvature at a point Q of the curve $\mathbf{r}$ at a time t. The interface motion given by (1.2) is called *anisotropic*. In the case $g, \beta \equiv$ const and $F \equiv 0$ the motion is called *isotropic*. Then we have (after normalizing constants) the simple relation

$$\mathcal{V} = k, \tag{1.3}$$

which means that the velocity of the curve in the inside normal direction is proportional to its curvature. The curve evolution governed by (1.3) is also called the "curve shortening flow." Gage and Hamilton show in [GH] that this problem, which starts from a *closed* and *convex* initial curve, is equivalent to the following initial value problem for a special partial differential equation (PDE).

Let the initial convex curve $\mathbf{r}_0$ be parametrized by θ, $k_0(\theta)$ be its curvature, and $\Omega = [0, 2\pi\nu]$, $\nu \in \mathbb{N}$. Assume that the function $k(\theta, t)\, \forall t \in [0, T), \theta \in \mathbb{R}$ satisfies the *curve shortening equation*, i.e.,

$$\partial_t k = k^2(k_{\theta\theta} + k), \tag{1.4}$$

the periodicity, and initial conditions

$$k(\theta, t) = k(\theta + 2\pi\nu, t), \tag{1.5}$$

$$k(\theta, 0) = k_0(\theta). \tag{1.6}$$

Then the flow $\mathbf{r}(\theta, t)$ of the curves that solves the problem (1.3) with the initial curve $\mathbf{r}_0$ is given uniquely up to translation by the formula

$$\mathbf{r}(\theta_0, t) = \mathbf{r}(0, t) - \int_0^{\theta_0} \frac{e^{i\theta}}{k(\theta, t)} d\theta. \tag{1.7}$$

The condition of closedness $\int_0^{2\pi\nu} \frac{e^{i\theta}}{k(\theta,t)} d\theta = 0$ is an invariant of the flow. By conservation of convexity $k(\theta, t) < 0 \, \forall t \in [0, T)$. Note that, by the convention introduced in [AG1], the curvature of strictly convex curves is negative. Papers [GH], [Gr], [ALa], [A1], [A2] describe many interesting properties of isotropic curve shortening; we will illustrate them in the section containing numerical computations (§7).

Equation (1.4) also serves as the model of a magnetic field in plasma physics. (There are Dirichlet instead of periodic boundary conditions.) The blow up of the solution in this case was studied from theoretical point of view by Friedman and McLeod in [FM]; numerics were done in [W], [Z].

Angenent and Gurtin [AG1] found the equation

$$\partial_t k = k^2 \left(\frac{gk - F}{\beta}\right)_{\theta\theta} + k^2 \left(\frac{gk - F}{\beta}\right) \tag{1.8}$$

for the dynamics of the curvature in the anisotropic case. This equation together with conditions (1.5), (1.6), and with $g(\theta) > 0$ describes the *anisotropic motion of the strictly convex interface*. That flow conserves convexity; the closedness condition is also an invariant (see [AG2]). Up to condition (H4), problem (1.8) may be transformed into the form (1.1) provided one puts

$$b(v) = -\frac{1}{v}, \quad A = D = \frac{g}{\beta}, \quad B = \left(\frac{g}{\beta}\right)_x, \quad G = -\frac{F}{\beta} - \left(\frac{F}{\beta}\right)_{xx}. \tag{1.9}$$

Using the transformation $\overline{b}(s) = b(s + c_1) - c_2$, where c_1, c_2 are constants, one of which is determined by the other, we can always satisfy (H4). The solutions of (1.1) with transform functions $\overline{b}$ and b, respectively, then correspond to each other uniquely. In this sense (1.1) is a model of the anisotropic motion of strictly convex phase interfaces.

The paper is organized as follows. In §2 we introduce two approximation schemes to solve (1.1). The convergence of the first approximation scheme is proved in §3. In §4 we discuss the regularization of the original problem (1.1). The convergence of the second approximation scheme is proved in §5. The error estimate analysis is presented in §6. In §7 we discuss the numerical results. It is well known that the isoperimetric ratio decreases to 1 in the isotropic time evolution of curves. We define an isoperimetric ratio generalization for the anisotropic time evolution. Our numerical experiments confirm decreasing to 1 of that *anisotropic isoperimetric ratio.*

2. Approximation schemes—the main theoretical results. Our main goal is to solve numerically the problem (1.1), especially in the case of anisotropic curve shortening (1.9). For this purpose we regularize the original problem in the following way. Let us consider the function $b_R(s)$ instead of $b(s)$, defined by

$$b_R(s) = \begin{cases} b(-R-\sigma) + b'(-R-\sigma)(s+R+\sigma), & s \le -R-\sigma, \\ r_1(s), & s \in (-R-\sigma, -R), \\ b(s), & s \in [-R, a - \frac{1}{R}], \\ r_2(s), & s \in (a - \frac{1}{R}, a - \frac{1}{R} + \sigma), \\ b(a - \frac{1}{R} + \sigma) + b'(a - \frac{1}{R} + \sigma)(s - a + \frac{1}{R} - \sigma), & s \ge a - \frac{1}{R} + \sigma, \end{cases} \tag{2.1}$$

where R, σ are real constants, $\sigma < \frac{1}{R}$. r_1 and r_2 are functions from $C^2(\mathbb{R})$ such that the whole function $b_R \in C^2(\mathbb{R})$ and inequalities $b'_R(-R-\sigma) \le b'_R(s) \le b'_R(a - \frac{1}{R} + \sigma)$ hold $\forall s \in \mathbb{R}$.

Now we define the *regularized problem*

$$\begin{aligned}\partial_t b_R(v) &= (Av_x)_x + (Bv)_x + Dv + G,\\ v(x,t) &= v(x+2\pi\nu, t),\\ v(x,0) &= v_0(x)\end{aligned} \tag{2.2}$$

for $x \in \mathbb{R}, t \in [0,T)$, $v_0(x)$ smooth function, under assumption (H1) and definition (2.1). Thus $b_R(s)$ is a C^2-increasing globally Lipschitz continuous function in $\mathbb{R}$ such that

$$\Gamma \geq b_R'(s) \geq \gamma > 0, \quad b_R(0) = 0. \tag{2.3}$$

Denote $I = (0,T)$, $\Omega = (0, 2\pi\nu)$, $Q_T = \Omega \times I$. Let $V = \{w \in W_2^1(0, 2\pi\nu) : w(0) = w(2\pi\nu)\}$ and assume that $v_0(x) \in V$.

DEFINITION 2.1. *The function $v \in L_2(I,V)$ with $\partial_t b_R(v) \in L_2(I,V^*)$ is called a weak solution of* (2.2) *iff $v(x,0) = v_0(x)$ and the identity*

$$\langle \partial_t b_R(v), \varphi\rangle + (Av_x, \varphi_x) + (Bv, \varphi_x) - (Dv, \varphi) = (G, \varphi) \tag{2.4}$$

holds $\forall\varphi \in V$ and for almost every (a.e.) $t \in I$. Replacing b_R by b we obtain the definition of a weak solution of the original problem (1.1).

A weak solution of problem (2.2) will be found using Rothe's method.

Let $n \in \mathbb{N}$, $\tau = \frac{T}{n}$, $t_i = i\tau$ for $i = 0, \dots, n$, $u_0 = v_0(x)$. For every $i = 1, \dots, n$, let $u_i \in V$ be the function such that $\forall\varphi \in V$

$$\left(b_R'(u_{i-1})\left(\frac{u_i - u_{i-1}}{\tau}\right), \varphi\right) + (Au_{ix}, \varphi_x) + (Bu_i, \varphi_x) - (Du_i, \varphi) = (G, \varphi). \tag{2.5}$$

From these functions we construct Rothe's functions

$$\begin{aligned} &u^{(n)}(t) = u_{i-1} + (t - t_{i-1})\frac{u_i - u_{i-1}}{\tau} \quad \text{for } t_{i-1} \leq t \leq t_i, i = 1, \dots, n,\\ &\overline{u}^{(n)}(t) = u_i \quad \text{for } t_{i-1} < t \leq t_i, i = 1, \dots, n, \quad \overline{u}^{(n)}(0) = u_0. \end{aligned} \tag{2.6}$$

THEOREM 2.1. *There exists a unique weak solution u of* (2.2) *such that $u^{(n)} \to u$ in $C(I, L_2)$; $\overline{u}^{(n)}(t) \rightharpoonup u(t)$, $u^{(n)}(t) \rightharpoonup u(t)$ in $V\, \forall t \in I$; $\partial_t u^{(n)} \rightharpoonup \partial_t u$ in $L_2(I, L_2)$ for the Rothe functions defined by* (2.6). *Moreover $u \in C(I, L_2) \cap L_\infty(I, V)$ and $\partial_t u \in L_2(I, L_2)$.*

The next theorem expresses the relation between the solutions of the regularized and the original problems with the bounded initial condition.

THEOREM 2.2. *Let u be a weak solution of* (2.2) *whose existence is guaranteed by the previous theorem. Let $u(x,0) \in L_\infty(\Omega) \cap V$ and*

$$-R + \delta < u(x,0) < a - \frac{1}{R} - \delta. \tag{2.7}$$

Then there exists $0 < T_1 < \infty$ so that

(i) *$u \in L_\infty(I, V) \cap L_\infty(Q_{T_1})$, and $Q_{T_1} = (0, T_1) \times \Omega$,*

(ii) *at the same time, u is a weak solution of* (1.1) *with initial condition $v_0(x) = u(x,0)$ in the time interval $(0, T_1)$.*

Remark 2.1. In Theorem 2.2 results concerning problem (2.2) obtained in Theorem 2.1 are transferred so as to obtain local results concerning the original problem (1.1). In this sense, the process (2.5) defines the *first approximation scheme* for solving problem (1.1).

The *second approximation scheme* is based on works [MNV] and [JK1] concerning numerical solution of the slow diffusion and Stefan-like problems and mainly on [JK2] describing the solution of elliptic–parabolic problems. This scheme is based on a nonstandard time discretization using the relaxation function by means of which we reduce the nonlinear original problem to an iterative solution of linear elliptic problems.

Let $n \in \mathbb{N}$, $\tau = \frac{T}{n}$, $t_i = i\tau$ for $i = 0, \dots, n$, $v_0 = v_0(x)$, $u_0 = v_0$. For $i = 1, \dots, n$ we look for functions $u_i \in V$ and $\mu_i \in L_\infty(\Omega)$ such that $\forall \varphi \in V$

$$(\mu_i(u_i - v_{i-1}), \varphi) + \tau(Au_{ix}, \varphi_x) + \tau(Bu_i, \varphi_x) - \tau(Du_i, \varphi) = \tau(G, \varphi), \tag{2.8}$$

satisfying the "convergence condition"

$$\alpha\frac{\gamma}{2} \le \mu_i \le \frac{b_R(v_{i-1} + \alpha(u_i - v_{i-1})) - b_R(v_{i-1})}{u_i - v_{i-1}}, \tag{2.9}$$

where $0 < \alpha < 1$. If $u_i = v_{i-1}$ then we replace the difference quotient by $\alpha b'_R(v_{i-1})$. The function v_i is obtained by the algebraic correction

$$b_R(v_i) := b_R(v_{i-1}) + \mu_i(u_i - v_{i-1}). \tag{2.10}$$

There are many possibilities to determine u_i, μ_i satisfying (2.8) and (2.9). If we choose $\alpha\frac{\gamma}{2} \le \mu_i \le \alpha\gamma$ then (2.8) and (2.9) are fulfilled. However, it is more interesting (from the numerical point of view) to choose μ_i very close to the difference quotient in (2.9). This can be done iteratively; see Remark 2.2.

From u_i, v_i, obtained in each time step of (2.8)–(2.10), the Rothe functions

$$u^{(n)}, v^{(n)}, \overline{u}^{(n)}, \overline{v}^{(n)} \tag{2.11}$$

are constructed (see (2.6)). Then we can prove the following theorem.

THEOREM 2.3. *Let* $\{u^{(n)}\}$, $\{v^{(n)}\}$, $\{\overline{u}^{(n)}\}$, $\{\overline{v}^{(n)}\}$ *be the sequences from* (2.11). *Then* $u^{(n)} \to u, v^{(n)} \to u, \overline{u}^{(n)} \to u, \overline{v}^{(n)} \to u$ *in* $L_2(Q_T)$, $\overline{u}^{(n)} \rightharpoonup u$ *in* $L_2(I, V)$, *where* u *is a weak solution of* (2.2).

Using techniques developed in [ALu] and [KHK], stronger convergence results can be proved.

THEOREM 2.4. *Let* $\{\overline{u}^{(n)}\}$ *be the sequence defined in* (2.11). *Then* $\overline{u}^{(n)} \to u$ *in* $L_2(I, V)$, *where* u *is a weak solution of* (2.2).

Remark 2.2. The question is how to find in a constructive way the couple u_i, μ_i simultaneously satisfying (2.8) and (2.9). We use iterations similar to the ones from [JK1], [JK2]:

(2.8')

$$(\mu_{i,k-1}(u_{i,k} - v_{i-1}), \varphi) + \tau(Au_{i,kx}, \varphi_x) + \tau(Bu_{i,k}, \varphi_x) - \tau(Du_{i,k}, \varphi) = \tau(G, \varphi),$$

$$\overline{\mu}_{i,k} = \frac{b_R(v_{i-1} + \alpha(u_{i,k} - v_{i-1})) - b_R(v_{i-1})}{u_{i,k} - v_{i-1}},$$

$$\mu_{i,k} := \overline{\mu}_{i,k} \text{ for } 1 \le k \le k_0, \quad \mu_{i,k} := \min\{\overline{\mu}_{i,k}, \mu_{i,k-1}\} \text{ for } k = k_0 + 1, \dots, \tag{2.9'}$$

starting with

$$\mu_{i,0} = b'_R(v_{i-1}). \tag{2.9''}$$

Note that if $u_{i,k} = v_{i-1}$ then we put $\overline{\mu}_{i,k} = \alpha b'_R(v_{i-1})$.

Generally ($b(s)$ is monotone) we are not able to prove the convergence of the sequence $\{\mu_{i,k}\}$ for $\mu_{i,k} \equiv \overline{\mu}_{i,k}$. By construction (2.9') with $k_0 \geq 1$, $\{\mu_{i,k}\}$ is forced to be monotone and hence convergent. In practical implementations k_0 can be chosen in accordance with the shape of $b(s)$. In our numerical realization (in case (1.9)) we have observed that $\mu_{i,k} \equiv \overline{\mu}_{i,k}$ are convergent and hence we have used k_0 sufficiently large. In practice the iterations very quickly converge to functions satisfying (2.8) and (2.9), and from theoretical point of view we have the following

THEOREM 2.5. *Let $1 \leq i \leq n$ be fixed, $\tau \leq \tau_0$, and $v_{i-1} \in L_\infty(\Omega)$. If $\{\mu_{i,k}\}$, $\{u_{i,k}\}$ are sequences from* (2.8') *and* (2.9') *then there exist $\mu \in L_\infty(\Omega)$, $w \in V$ such that*

$$\mu_{i,k} \to \mu =: \mu_i \text{ in } L_p(\Omega) \ (\forall p > 1), \quad u_{i,k} \to w =: u_i \text{ in } C^{0,\delta}(\overline{\Omega}) \ \left(\delta < \frac{1}{2}\right)$$

for $k \to \infty$, where μ_i, u_i satisfy (2.8) *and* (2.9).

The next theorem concerns the "error estimate" for approximation scheme (2.8)–(2.10).

THEOREM 2.6. *Let u be a weak solution of* (2.2) *and $\overline{u}^{(n)}$ the sequence of Rothe's functions obtained by the approximation scheme* (2.8)–(2.10). *Then the following estimate holds:*

$$\int_I |\overline{u}^{(n)}(s) - u(s)|_2^2 ds + \left| \int_I (\overline{u}^{(n)}(s) - u(s))_x ds \right|_2^2 \leq C\tau. \tag{2.12}$$

The function u in the previous theorem is an arbitrary weak solution of the regularized problem (2.2). Its uniqueness is proved using (2.12). By standard arguments, the original sequences given in (2.11) and (2.6), respectively, converge to this solution. Combining these results with Theorem 2.2 we can formulate the next theorem concerning the solution of the original problem (1.1).

THEOREM 2.7. *Let $v_0 \in L_\infty(\Omega) \cap V$ satisfy* (2.7). *Then there exist $T_1 < \infty$ and function $u \in C(I, L_2) \cap L_\infty(I, V) \cap L_\infty(Q_{T_1})$ with $\partial_t u \in L_2(I, L_2)$, $I = (0, T_1)$, which is the unique bounded solution of the problem* (1.1). *For $\overline{u}^{(n)}$, given by approximation scheme* (2.8)–(2.10), *we have $\overline{u}^{(n)} \to u$, $b(\overline{u}^{(n)}) \to b(u)$ in $L_2(I, V)$ and* (2.12).

3. Convergence of the approximation scheme (2.5).

Remark 3.1. We denote the scalar product in $L_2(\Omega)$ by $(\cdot, \cdot)$ and duality between V and V^* by $\langle \cdot, \cdot \rangle$. We use symbols $|\cdot|_2$, $\|\cdot\|$, $\|\cdot\|_*$, $|\cdot|_\infty$, $|\cdot|_p$ for norms in $L_2(\Omega)$, V, V^*, $L_\infty(\Omega)$, and $L_p(\Omega)$, respectively. By $\to$, $\rightharpoonup$, we mean strong and weak convergence; the Landau symbol $o(1)$ represents a term c_n satisfying $c_n \to 0$ for $n \to \infty$. C denotes the generic positive constant, ε small positive number, and $C(\varepsilon)$ term dependent on ε^{-1}. Let us denote $\delta u_i = \frac{u_i - u_{i-1}}{\tau}$ and $\delta \overline{u}^{(n)}(t) = \frac{\overline{u}^{(n)}(t) - \overline{u}^{(n)}(t-\tau)}{\tau}$.

LEMMA 3.1. *There exists $\tau_0 > 0$ such that $\forall n \in \mathbb{N}$ satisfying $\tau = \frac{T}{n} < \tau_0$ we have the following: for every $i = 1, \dots, n$ there exists unique $u_i \in V$ satisfying* (2.5) *and the following a priori estimates hold:*

$$\sum_{i=1}^n |\delta u_i|_2^2 \tau \leq C, \quad \max_{1 \leq i \leq n} \|u_i\| \leq C, \quad \sum_{i=1}^n |u_{ix} - u_{i-1x}|_2^2 \leq C. \tag{3.1}$$

Proof. We put $\varphi = u_i$ into (2.5) and use the estimates

$$\begin{aligned} &(Au_{ix}, u_{ix}) \geq q|u_{ix}|_2^2, \quad |(Bu_i, u_{ix})| \leq \varepsilon |u_{ix}|_2^2 + C(\varepsilon)|u_i|_2^2, \\ &|(Du_i, u_i)| \leq C|u_i|_2^2, \quad |(G, u_i)| \leq \varepsilon |G|_2^2 + C(\varepsilon)|u_i|_2^2. \end{aligned} \tag{3.2}$$

The Lax–Milgram theorem guarantees the existence of the unique $u_i \in V$ satisfying (2.5).

Let us put $\varphi = u_i - u_{i-1}$ into (2.5) and sum the equations for $i = 1, \dots, j, j \le n$. Using (2.3), we obtain the following inequality:

$$\begin{aligned}\gamma \sum_{i=1}^{j} |\delta u_i|_2^2 \tau + \sum_{i=1}^{j} (Au_{ix}, u_{ix} - u_{i-1x}) + \sum_{i=1}^{j} (Bu_i, u_{ix} - u_{i-1x})\\ - \sum_{i=1}^{j} (Du_i, u_i - u_{i-1}) \le \sum_{i=1}^{j} (G, u_i - u_{i-1}).\end{aligned} \tag{3.3}$$

Using Abel's summation, Young's inequality, (H1), and (2.3) we estimate

$$\sum_{i=1}^{j} (Au_{ix}, u_{ix} - u_{i-1x}) \ge \frac{q}{2} \sum_{i=1}^{j} |u_{ix} - u_{i-1x}|_2^2 + \frac{q}{2} |u_{jx}|_2^2 - C,$$

$$\begin{aligned}\sum_{i=1}^{j} (Bu_i, u_{ix} - u_{i-1x}) &= - \sum_{i=1}^{j} (B(u_i - u_{i-1}), u_{i-1\,x}) + (Bu_j, u_{jx}) - (Bu_0, u_{0x})\\ &\le \varepsilon |u_{jx}|_2^2 + C(\varepsilon)|u_j|_2^2 + C(\varepsilon) \sum_{i=1}^{j} |u_{ix}|_2^2 \tau + \varepsilon \sum_{i=1}^{j} |\delta u_i|_2^2 \tau + C,\end{aligned}$$

$$\left| \tau \sum_{i=1}^{j} \left(Du_i, \frac{u_i - u_{i-1}}{\tau} \right) \right| \le \varepsilon \sum_{i=1}^{j} |\delta u_i|_2^2 \tau + C(\varepsilon) \sum_{i=1}^{j} |u_i|_2^2 \tau,$$

$$\left| \tau \sum_{i=1}^{j} \left(G, \frac{u_i - u_{i-1}}{\tau} \right) \right| \le \varepsilon \sum_{i=1}^{j} |\delta u_i|_2^2 \tau + C(\varepsilon).$$

In the previous inequalities we used an estimate for $|u_j|_2^2$. The parabolic term gives us

$$\begin{aligned}&\frac{\gamma}{2} |u_j|_2^2 - C + \frac{1}{2} \sum_{i=1}^{j} \int_\Omega (b_R'(u_{i-1}) - b_R'(u_i)) u_i^2 dx\\ &\le \frac{1}{2} \sum_{i=1}^{j} (b_R'(u_{i-1})(u_i - u_{i-1}), u_i - u_{i-1}) + \frac{1}{2} \sum_{i=1}^{j} (b_R'(u_{i-1}) u_i, u_i)\\ &- \frac{1}{2} \sum_{i=1}^{j} (b_R'(u_{i-1}) u_{i-1}, u_{i-1}) = \sum_{i=1}^{j} (b_R'(u_{i-1})(u_i - u_{i-1}), u_i)\\ &\le \delta \sum_{i=1}^{j} |\delta u_i|_2^2 \tau + C(\delta) \sum_{i=1}^{j} |u_i|_2^2 \tau.\end{aligned}$$

The regularized function b_R satisfies the condition $|b_R''(z)| \le \frac{C_1}{1+|z|^s}$, with $C_1 > 0$ and $s > 1$. Then there exists $K > 0$ such that (see [K3, Lem. 3])

$$\begin{aligned}&\left| \int_\Omega (b_R'(u_{i-1}) - b_R'(u_i)) u_i^2 dx \right| = \left| \int_\Omega (u_{i-1} - u_i) \int_0^1 b_R''(u_i + \omega(u_{i-1} - u_i)) d\omega u_i^2 dx \right|\\ &\le C_1 \int_\Omega |u_i - u_{i-1}| \int_0^1 \frac{|u_i|}{1 + |u_i + \omega(u_{i-1} - u_i)|^s} d\omega |u_i| dx \le C_1 K \int_\Omega |u_i - u_{i-1}||u_i| dx.\end{aligned}$$

Estimating the last term $\int_\Omega |u_i - u_{i-1}||u_i| \le \delta |\delta u_i|_2^2 \tau + C(\delta)|u_i|_2^2 \tau$ we obtain

$$|u_j|_2^2 \le C + C(\delta) \sum_{i=1}^{j} |u_i|_2^2 \tau + \delta \sum_{i=1}^{j} |\delta u_i|_2^2 \tau.$$

Applying all the previous estimates in (3.3) we deduce (for C sufficiently large)

$$\sum_{i=1}^{j} |\delta u_i|_2^2 \tau + |u_{jx}|_2^2 + \sum_{i=1}^{j} |u_{ix} - u_{i-1x}|_2^2 \le C + C \sum_{i=1}^{j} |u_{ix}|_2^2 \tau + C \sum_{i=1}^{j} \left(\sum_{l=1}^{i} |\delta u_l|_2^2 \tau \right) \tau.$$

Here, we apply Gronwall's argument and obtain desired a priori estimates. □

Proof of Theorem 2.1. For the Rothe functions defined in (2.6), we can rewrite a priori estimates (3.1) in the following form:

$$\int_I |\partial_t u^{(n)}(t)|_2^2 dt \le C, \quad \max_{t \in [0,T]} \|\overline{u}^{(n)}(t)\| \le C \; \forall n \ge n_0.$$

Then by standard arguments (see [K1, Lem. 1.3.13]) we obtain $u \in C(I, L_2) \cap L_\infty(I, V)$ with $\partial_t u \in L_2(I, L_2)$ such that in a sense of subsequences (a subsequence of $\{n\}$ we denote again by $\{n\}$) $u^{(n)} \to u$ in $C(I, L_2)$; $\overline{u}^{(n)}(t) \rightharpoonup u(t)$, $u^{(n)}(t) \rightharpoonup u(t)$ in V $\forall t \in I$; $\partial_t u^{(n)} \rightharpoonup \partial_t u$ in $L_2(I, L_2)$. Lemma 3.1 implies

$$\int_I |\overline{u}^{(n)}(t) - \overline{u}^{(n)}(t-\tau)|_2^2 dt \le \frac{C}{n^2}, \quad \int_I |\overline{u}^{(n)}(t) - u^{(n)}(t)|_2^2 dt \le \frac{C}{n^2}.$$

Now taking the limit for $n \to \infty$ in

$$\begin{aligned} &\int_J (\partial_t u^{(n)}(t), b'_R(\overline{u}^{(n)}(t-\tau))\varphi) dt + \int_J (A\overline{u}_x^{(n)}(t), \varphi_x) dt \\ &+ \int_J (B\overline{u}^{(n)}(t), \varphi_x) dt - \int_J (D\overline{u}^{(n)}(t), \varphi) dt = \int_J (G, \varphi) dt, \end{aligned}$$

where J is a subinterval of I, we obtain that u satisfies (2.4). The uniqueness of a weak solution will be guaranteed by Theorem 2.6. The convergence of the original sequences $\{u^{(n)}\}$, $\{\overline{u}^{(n)}\}$ then follows from the uniqueness argument. □

4. Relation between regularized and original problems.

Proof of Theorem 2.2. Let us denote

$$\gamma(R) =: b'(-R - \sigma), \quad \Gamma(R) =: b'\left(a - \frac{1}{R} + \sigma\right) \quad \text{for } R \ge R_0.$$

First we shall prove part (i). We obtained the solution u from the theorem by solving identity (2.5) for $i = 1, \dots, n$. Due to (2.3) and $u_0 \in L_\infty(\Omega)$, we have that $u_i \in L_\infty(\Omega)$ for $i = 1, \dots, n$ (see [L], [LSU]). Now we shall derive the L_∞-estimate for u_i. In what follows we use similar ideas to those in [K2]. Let us consider u_i^p, where p is odd, as a test function in the identity (2.5). (Since $u_i \in L_\infty(\Omega)$ we have $u_i^p \in V$.) Using Young's inequality we obtain

$$\begin{aligned} &\int_\Omega b'_R(u_{i-1}) u_i^{p+1} + p\tau \int_\Omega A u_{ix}^2 u_i^{p-1} \le \int_\Omega b'_R(u_{i-1}) u_{i-1} u_i^p \\ &+ \tau \left| \int_\Omega B u_i (u_i^p)_x \right| + \tau \int_\Omega |D| u_i^{p+1} + \tau \frac{\varepsilon^{p+1}}{p+1} \int_\Omega G^{p+1} + \tau \frac{p}{\varepsilon^{\frac{p+1}{p}} (p+1)} \int_\Omega u_i^{p+1}. \end{aligned} \tag{4.1}$$

Using periodicity, Young's inequality, and (H1) we estimate

$$\begin{aligned} \left| \int_\Omega B u_i (u_i^p)_x \right| &= \left| [B u_i^{p+1}]_0^{2\pi\nu} - \int_\Omega (B u_i)_x u_i^p \right| \le \int_\Omega |B_x| u_i^{p+1} + \int_\Omega |B||u_i|^p |u_{ix}| \\ &\le \int_\Omega |B_x| u_i^{p+1} + \frac{1}{2} \int_\Omega A u_i^{p-1} u_{ix}^2 + C \int_\Omega u_i^{p+1}. \end{aligned}$$

Applying it in (4.1) together with $b'_R(u_{i-1}) \geq \gamma(R) > 0$ and (H1) we obtain

$$\int_\Omega b'_R(u_{i-1})u_i^{p+1} + \left(p - \frac{1}{2}\right)\tau \int_\Omega Au_{ix}^2 u_i^{p-1} \leq \int_\Omega b'_R(u_{i-1})u_{i-1}u_i^p$$
$$+\tau K \int_\Omega b'_R(u_{i-1})u_i^{p+1} + \tau \frac{\varepsilon^{p+1}}{p+1}\int_\Omega G^{p+1},$$

where the constant K depends on $|B_x|_\infty, |D|_\infty, |B|_\infty, |A|_\infty, R, \varepsilon$. Hence we have

$$(1 - K\tau)\int_\Omega b'_R(u_{i-1})u_i^{p+1} \leq \int_\Omega b'_R(u_{i-1})u_{i-1}u_i^p + \tau\frac{\varepsilon^{p+1}}{p+1}\int_\Omega G^{p+1}.$$

Then for τ sufficiently small ($\frac{T}{n} = \tau \leq \frac{1}{K}$) we obtain (for $\varepsilon_n = \frac{2K^2T}{n}$)

$$\int_\Omega b'_R(u_{i-1})u_i^{p+1} \leq (1 + (K + \varepsilon_n)\tau)\left[\int_\Omega b'_R(u_{i-1})u_{i-1}u_i^p + \tau\frac{\varepsilon^{p+1}}{p+1}\int_\Omega G^{p+1}\right].$$

Using Young's inequality for the first term on the right-hand side we deduce

$$\int_\Omega b'_R(u_{i-1})u_i^{p+1} \leq \frac{(1 + (K+\varepsilon_n)\tau)^{p+1}}{p+1}\int_\Omega b'_R(u_{i-1})u_{i-1}^{p+1}$$
$$+\frac{p}{p+1}\int_\Omega b'_R(u_{i-1})u_i^{p+1} + (1 + (K+\varepsilon_n)\tau)\tau\frac{\varepsilon^{p+1}}{p+1}\int_\Omega G^{p+1}.$$

Now, using the estimates $\frac{\gamma(R)}{\Gamma(R)} \leq b'_R(u_{i-1}) \leq \frac{\Gamma(R)}{\gamma(R)}$ we obtain the recurrent inequality

$$\int_\Omega u_i^{p+1} \leq (1 + (K+\varepsilon_n)\tau)^{p+1}\left(\frac{\Gamma(R)}{\gamma(R)}\right)^2\left[\int_\Omega u_{i-1}^{p+1} + \tau\varepsilon^{p+1}\int_\Omega G^{p+1}\right]$$

and hence

$$\int_\Omega u_i^{p+1} \leq (1 + (K+\varepsilon_n)\tau)^{(p+1)i}\left(\frac{\Gamma(R)}{\gamma(R)}\right)^{2i}\left[\int_\Omega u_0^{p+1} + i\tau\varepsilon^{p+1}\int_\Omega G^{p+1}\right].$$

Here, we take the $(p+1)$th root and let $p \to \infty$. Then (because $(1+x)^i \leq e^{ix}$)

$$|u_i|_\infty \leq (1 + (K+\varepsilon_n)\tau)^i(|u_0|_\infty + \varepsilon|G|_\infty) \leq e^{(K+\varepsilon_n)t_i}(|u_0|_\infty + \varepsilon|G|_\infty). \tag{4.2}$$

Due to Theorem 2.1, the functions $u^{(n)}$ from (2.6) converge to the solution u in $C(I, L_2)$. Taking limit for $n \to \infty$ we derive from (4.2)

$$|u(t)|_\infty \leq e^{Kt}(|u_0|_\infty + \varepsilon|G|_\infty) \ \forall t \in I, \tag{4.3}$$

where constant K depends on $|B_x|_\infty, |D|_\infty, |B|_\infty, |A|_\infty, R, \varepsilon$. Assertion (i) of the theorem follows from (4.3). At the same time we proved that if $u(x, 0)$ satisfies the left inequality in (2.7) then for ε sufficiently small, there exists $0 < T_2 < \infty$ such that

$$-R \leq u(x,t) \quad \text{for a.e. } x \in \Omega, t \in (0, T_2). \tag{4.4}$$

Now, we shall look for the L_∞-estimate for the function $b_R(u)$ to obtain the similar result concerning the right inequality in (2.7). Due to (2.3) a weak solution u satisfies

$$\langle b_R'(u)\partial_t u, \varphi\rangle + (Au_x, \varphi_x) + (Bu, \varphi_x) - (Du, \varphi) = (G, \varphi) \quad \forall \varphi \in V. \tag{4.5}$$

From the previous part of the proof (where we derived that $u \in L_\infty(Q_{T_2})$), it follows that $b_R^p(u) \in V$ for $t \in (0, T_2)$. Thus we can put $\varphi = b_R^p(u)$, where p is odd, into (4.5) and integrate the identity over $(0, t), t \le T_2$. Using integration *per partes* in the convection term and considering the periodicity we obtain

$$\begin{aligned}
&\frac{1}{p+1}\left[\int_\Omega |b_R(u(t))|^{p+1} - \int_\Omega |b_R(u(0))|^{p+1}\right] + p\int_0^t\int_\Omega Au_x^2 b_R^{p-1}(u)b_R'(u) \\
&\le \int_0^t\int_\Omega |B_x||u||b_R(u)|^p + \int_0^t\int_\Omega |B||u_x||b_R(u)|^p + \int_0^t\int_\Omega |D||u||b_R(u)|^p \\
&+\frac{1}{p+1}\int_0^t\int_\Omega \varepsilon^{p+1}|G|^{p+1} + C(\varepsilon)\frac{p}{p+1}\int_0^t\int_\Omega |b_R(u)|^{p+1}.
\end{aligned} \tag{4.6}$$

The second term on the right-hand side can be estimated (for ε sufficiently small)

$$\begin{aligned}
\int_0^t\int_\Omega |B||u_x||b_R(u)|^p &\le \varepsilon\int_0^t\int_\Omega |B|^2u_x^2b_R^{p-1}(u) + C(\varepsilon)\int_0^t\int_\Omega b_R^{p+1}(u) \\
&\le \frac{1}{\gamma(R)}\int_0^t\int_\Omega Au_x^2b_R^{p-1}(u)b_R'(u) + C\int_0^t\int_\Omega b_R^{p+1}(u).
\end{aligned} \tag{4.7}$$

Because $|u| \le \frac{|b_R(u)|}{\gamma(R)}$ we have

$$\int_0^t\int_\Omega (|B_x| + |D|)|u||b_R(u)|^p \le \frac{(|B_x|_\infty + |D|_\infty)}{\gamma(R)}\int_0^t\int_\Omega b_R^{p+1}(u). \tag{4.8}$$

Applying (4.7) and (4.8) in (4.6) for $p > \frac{1}{\gamma(R)}$ we obtain

$$\int_\Omega |b_R(u(t))|^{p+1} \le L(p+1)\int_0^t\int_\Omega |b_R(u)|^{p+1} + T\int_\Omega \varepsilon^{p+1}|G|^{p+1} + \int_\Omega |b_R(u(0))|^{p+1},$$

where the constant L depends on $|B_x|_\infty, |D|_\infty, |B|_\infty, |A|_\infty, R, \varepsilon$. Using Gronwall's lemma and taking the $(p+1)$th root with $p \to \infty$ we obtain

$$|b_R(u(t))|_\infty \le e^{Lt}(|b_R(u(0))|_\infty + \varepsilon|G|_\infty) \quad \forall t \in (0, T_2). \tag{4.9}$$

Let $u(x, 0)$ satisfy the right inequality in (2.7). Then $|b_R(u(0))|_\infty < b_R(a - \frac{1}{R}) - \delta_1$ for some $\delta_1 > 0$. For ε sufficiently small, it follows from (4.9) that there exists $0 < T_3 < \infty$ such that $b_R(u(x,t)) \le b_R(a - \frac{1}{R})$ for a.e. $x \in \Omega, t \in (0, T_3)$. Because b_R is strictly increasing, we have

$$u(x,t) \le a - \frac{1}{R} \quad \text{for a.e. } x \in \Omega, t \in (0, T_3). \tag{4.10}$$

Let us set $T_1 = \min\{T_2, T_3\}$. Owing to the fact that the functions b and b_R are the same for arguments satisfying (4.4) and (4.10) simultaneously, we have that solution u of problem (2.2) is at the same time a solution of the original problem (1.1) in the interval $[0, T_1)$. □

5. Convergence of the approximation scheme (2.8)–(2.10).

Proof of Theorem 2.3. First we shall derive a priori estimates for the functions u_i, v_i. For this purpose we use the techniques from [JK2]. Let us put $\varphi = u_i$ into (2.8) for $i = 1, \dots, j$ and sum these identities. Using estimates (3.2) we obtain (for C sufficiently large)

$$\sum_{i=1}^{j}(\mu_i(u_i - v_{i-1}), u_i) + \sum_{i=1}^{j} |u_{ix}|_2^2\tau \le C + C\sum_{i=1}^{j} |u_i|_2^2\tau. \tag{5.1}$$

Using (2.10) we can rewrite the first term in (5.1) in the following way:

$$\begin{aligned} J_1 &= \sum_{i=1}^{j}\left(b_R(v_i) - b_R(v_{i-1}), (b_R(v_i) - b_R(v_{i-1}))\frac{1}{\mu_i} + v_{i-1}\right) \\ &= \sum_{i=1}^{j}\int_\Omega \frac{1}{\mu_i}(b_R(v_i) - b_R(v_{i-1}))^2 + \sum_{i=1}^{j}(b_R(v_i) - b_R(v_{i-1}), v_i) \\ &\quad - \sum_{i=1}^{j}(b_R(v_i) - b_R(v_{i-1}), v_i - v_{i-1}). \end{aligned} \tag{5.2}$$

Let us define $\Phi_R(s) = \int_0^s b_R(z)dz$, $\mathcal{B}_R(s) = b_R(s)s - \Phi_R(s)$. Because b_R is a continuous increasing function, the following inequality holds: $\Phi_R(v_i) - \Phi_R(v_{i-1}) \ge b_R(v_{i-1})(v_i - v_{i-1})$. It can be used to estimate the second summand in (5.2). We obtain

$$\begin{aligned} &\sum_{i=1}^{j}(b_R(v_i) - b_R(v_{i-1}), v_i) = (b_R(v_j), v_j) - (b_R(v_0), v_0) - \sum_{i=1}^{j}(b_R(v_{i-1}), v_i - v_{i-1}) \\ &\ge (b_R(v_j), v_j) - (b_R(v_0), v_0) - \sum_{i=1}^{j}\int_\Omega \Phi_R(v_i) - \Phi_R(v_{i-1}) = \int_\Omega \mathcal{B}_R(v_j) - \int_\Omega \mathcal{B}_R(v_0). \end{aligned}$$

The algebraic correction (2.10) and the convergence condition (2.9) imply

$$|b_R(v_i) - b_R(v_{i-1})| = |\mu_i(u_i - v_{i-1})| \le |b_R(v_{i-1} + \alpha(u_i - v_{i-1})) - b_R(v_{i-1})|.$$

Hence and from strict monotonicity of b_R we have

$$|v_i - v_{i-1}| \le |\alpha(u_i - v_{i-1})| = \frac{\alpha}{\mu_i}|b_R(v_i) - b_R(v_{i-1})|,$$

which implies

$$\left|\sum_{i=1}^{j}(b_R(v_i) - b_R(v_{i-1}), v_i - v_{i-1})\right| \le \alpha\sum_{i=1}^{j}\int_\Omega \frac{1}{\mu_i}(b_R(v_i) - b_R(v_{i-1}))^2.$$

Using it and the estimate $\mathcal{B}_R(s) = b_R(s)s - \int_0^s b_R(z)dz \ge \gamma s^2 - C$ in (5.2) we obtain

$$\begin{aligned} J_1 &\ge (1-\alpha)\sum_{i=1}^{j}\int_\Omega \frac{1}{\mu_i}(b_R(v_i) - b_R(v_{i-1}))^2 + \int_\Omega \mathcal{B}_R(v_j) - \int_\Omega \mathcal{B}_R(v_0) \\ &\ge \frac{1}{\Gamma}(1-\alpha)\sum_{i=1}^{j}|b_R(v_i) - b_R(v_{i-1})|_2^2 + \gamma|v_j|_2^2 - C. \end{aligned} \tag{5.3}$$

From (2.9) and (2.10) we have

$$\sum_{i=1}^{j} |u_i|_2^2 \tau \le 2\tau \frac{C}{\gamma^2} \sum_{i=1}^{j} |b_R(v_i) - b_R(v_{i-1})|_2^2 + 2\tau \sum_{i=1}^{j} |v_i|_2^2 + C. \tag{5.4}$$

Using (5.3) and (5.4) in (5.1) and applying Gronwall's argument we obtain a priori estimates

$$\max_{1\le i\le n} |v_i|_2^2 + \sum_{i=1}^{j} |u_{ix}|_2^2 \tau + \sum_{i=1}^{j} |b_R(v_i) - b_R(v_{i-1})|_2^2 \le C. \tag{5.5}$$

As a consequence of (5.5), (2.3), and (2.10) we have

$$\begin{gathered} \sum_{i=1}^{n} |u_i - v_{i-1}|_2^2 \le C, \quad \sum_{i=1}^{n} |v_i - v_{i-1}|_2^2 \le C, \\ \sum_{i=1}^{n} |u_i - v_i|_2^2 \le C, \quad \sum_{i=1}^{n} |u_i - u_{i-1}|_2^2 \le C, \quad |u_i|_2^2 \le C. \end{gathered} \tag{5.6}$$

The estimates (5.5) and (5.6) imply for the Rothe functions defined in (2.11) the following properties (holding uniformly for $n \ge n_0$):

$$\begin{gathered} \int_0^{T-\tau} |\overline{u}^{(n)}(t+\tau) - \overline{u}^{(n)}(t)|_2^2 dt \le \frac{C}{n}, \quad \int_I |\overline{u}_x^{(n)}(t)|_2^2 dt \le C, \\ |\overline{u}^{(n)}(t)|_2^2 \le C, \quad |\overline{v}^{(n)}(t)|_2^2 \le C \ \forall t \in I. \end{gathered} \tag{5.7}$$

Let $y_0 > 0$ be a sufficiently small real number. Let $\Omega^* \subset \Omega$ be an open subinterval of Ω, such that $\overline{\Omega^*} \subset \Omega$ and $x + y \in \Omega \, \forall x \in \Omega^* \ \forall y, |y| \le y_0$. Easily we obtain

$$\int_{\Omega^*} (\overline{u}^{(n)}(t, x+y) - \overline{u}^{(n)}(t,x))^2 dx \le |y| \int_{\Omega} (\overline{u}_x^{(n)}(t,z))^2 dz,$$

which together with the second inequality in (5.7) implies (for $|y| < y_0$)

$$\int_I \int_{\Omega^*} (\overline{u}^{(n)}(t, x+y) - \overline{u}^{(n)}(t,x))^2 dx dt \le |y| \int_I |\overline{u}_x^{(n)}(t)|_2^2 \le C|y|. \tag{5.8}$$

Then from the first inequality in (5.7) and (5.8) we have

$$\int_0^{T-z} \int_{\Omega^*} (\overline{u}^{(n)}(t+z, x+y) - \overline{u}^{(n)}(t,x))^2 dx dt \le C(z + |y|), \tag{5.9}$$

which holds uniformly due to n and $\forall\, 0 < z < z_0$, $|y| < y_0$. It follows from the third inequality in (5.7) and (5.9) that the sequence $\{\overline{u}^{(n)}(x,t)\}_{n=1}^{\infty}$ is compact in $L_2(Q_T)$. Thus there exists a subsequence of $\{\overline{u}^n\}_{n=1}^{\infty}$ which converges in $L_2(Q_T)$ to a function u. (We denote this subsequence again by $\overline{u}^{(n)}$.) Using (5.5) and (5.6) we obtain the following convergence results (in the sense of subsequences):

$$u^{(n)} \to u, v^{(n)} \to u, \overline{u}^{(n)} \to u, \overline{v}^{(n)} \to u \text{ in } L_2(Q_T), \quad \overline{u}^{(n)} \rightharpoonup u \text{ in } L_2(I, V),$$

$$b_R(u^{(n)}) \to b_R(u), b_R(v^{(n)}) \to b_R(u), b_R(\overline{u}^{(n)}) \to b_R(u), b_R(\overline{v}^{(n)}) \to b_R(u) \text{ in } L_2(Q_T). \tag{5.10}$$

Due to (2.8), (2.10), and (H1) we have

$$\sup_{\substack{\|\varphi\|\le 1\\ \varphi\in V}} \left| \left(\frac{b_R(v_i) - b_R(v_{i-1})}{\tau}, \varphi \right) \right| \le C\|u_i\|,$$

which implies $\|\delta b_R(v_i)\|_* \le C\|u_i\|$. Owing to a priori estimates (5.5) and (5.6) we obtain

$$\|\delta b_R(\overline{v}^{(n)})\|^2_{L_2(I,V^*)} = \int_I \|\delta b_R(\overline{v}^{(n)})\|^2_* dt = \sum_{i=1}^n \|\delta b_R(v_i)\|^2_* \tau \le C \sum_{i=1}^n \|u_i\|^2 \tau \le C,$$

and reflexivity of $L_2(I, V^*)$ implies $\delta b_R(\overline{v}^{(n)}) \rightharpoonup \chi$ in $L_2(I, V^*)$. Hence and from the ninth convergence result stated in (5.10) we obtain $\chi \equiv \partial_t b_R(u)$. Integrating the identity (2.8) over a subinterval $J \subset I$ we have $\forall \varphi \in V$

$$\begin{aligned} \int_J (\delta b_R(\overline{v}^{(n)}(t)), \varphi) dt + \int_J (A\overline{u}_x^{(n)}(t), \varphi_x) dt + \int_J (B\overline{u}^{(n)}(t), \varphi_x) dt \\ - \int_J (D\overline{u}^{(n)}(t), \varphi) dt = \int_J (G, \varphi) dt. \end{aligned} \tag{5.11}$$

Here taking limit for $n \to \infty$ we conclude

$$\langle \partial_t b_R(u), \varphi \rangle + (Au_x, \varphi_x) + (Bu, \varphi_x) - (Du, \varphi) = (G, \varphi) \ \forall \varphi \in V \text{ a.e. } t \in I,$$

so u is a weak solution of problem (2.2) in the sense of Definition 2.1. The convergence of the original sequences $\{u^{(n)}\}$, $\{v^{(n)}\}$, $\{\overline{u}^{(n)}\}$, $\{\overline{v}^{(n)}\}$ follows from the uniqueness argument. □

Proof of Theorem 2.4. Let us put $\varphi = \overline{u}^{(n)} - u$ and estimate the terms in (5.11) for $J \equiv (0, t)$. Due to (5.3), (5.10), and Fatou's lemma we obtain

$$\underline{\lim} \int_0^t (\delta b_R(\overline{v}^{(n)}), \overline{u}^{(n)}) ds \ge \int_\Omega \mathcal{B}_R(u(t)) dx - \int_\Omega \mathcal{B}_R(u(0)) dx$$

for a.e. $t \in I$. We have

$$\int_0^t (\delta b_R(\overline{v}^{(n)}), u) ds \to \int_0^t \langle \partial_t b_R(u), u \rangle = \int_\Omega \mathcal{B}_R(u(t)) dx - \int_\Omega \mathcal{B}_R(u(0)) dx,$$

where the last equality holds due to [ALu, Lem. 1.5]. Using the previous two estimates we have

$$\underline{\lim} \int_0^t (\delta b_R(\overline{v}^{(n)}), \overline{u}^{(n)} - u) ds \ge 0.$$

We estimate the second term in (5.11) in the following way:

$$\begin{aligned} \int_0^t (A\overline{u}_x^{(n)}, (\overline{u}^{(n)} - u)_x) ds = \int_0^t (A(\overline{u}^{(n)} - u)_x, (\overline{u}^{(n)} - u)_x) ds \\ + \textstyle\int_0^t (Au_x, (\overline{u}^{(n)} - u)_x) ds \ge q \int_0^t |(\overline{u}^{(n)} - u)_x|_2^2 - o(1), \end{aligned}$$

since $\overline{u}^{(n)} \rightharpoonup u$ in $L_2(I, V)$. In the third term we use integration *per partes* and obtain

$$\left| \int_0^t (B\overline{u}^{(n)}, (\overline{u}^{(n)} - u)_x) ds \right| = \left| \int_0^t ((B\overline{u}^{(n)})_x, \overline{u}^{(n)} - u) ds \right| = o(1)$$

because $\overline{u}^{(n)} \to u$ in $L_2(Q_T)$. It is for the same reason that the other terms in (5.11) are also of type $o(1)$. Thus we have the required result. □

Proof of Theorem 2.5. The monotonicity of the sequence $\{\mu_{i,k}\}$ from (2.9') for $k \geq k_0$ implies $\mu_{i,k} \to \mu$ pointwise in Ω. Since $\alpha\frac{\gamma}{2} \leq \mu_{i,k} \leq \Gamma$ we have $\mu_{i,k} \to \mu$ in $L_p(\Omega)\,\forall p > 1$. On the other hand, if $v_{i-1} \in L_\infty(\Omega)$, then the sequence $\{u_{i,k}\}$ is bounded in the norm of V and hence bounded in $C^{0,\overline{\delta}}(\Omega)$ with $\overline{\delta} = \frac{1}{2}$; see [KJF].

If we subtract (2.8') for $k = r, s$ and use the test function $\varphi = u_{i,r} - u_{i,s}$ we obtain

$$\alpha\frac{\gamma}{2}|u_{i,r} - u_{i,s}|_2^2 + \tau q|(u_{i,r} - u_{i,s})_x|_2^2 \leq \int_\Omega |\mu_{i,r-1} - \mu_{i,s-1}||u_{i,r} - u_{i,s}||u_{i,s} - v_{i-1}|dx$$
$$+ \tau\varepsilon C|(u_{i,r} - u_{i,s})_x|_2^2 + \tau C(\varepsilon)|u_{i,r} - u_{i,s}|_2^2.$$

Then for ε sufficiently small and $\tau \leq \tau_0$, from generalized Hölder inequality and from the imbedding of V to $L_q(\Omega)(\forall q > 1)$, we have

$$|u_{i,r} - u_{i,s}|_2^2 \leq C|u_{i,r} - u_{i,s}|_2|\mu_{i,r-1} - \mu_{i,s-1}|_4(|u_{i,s}|_4 + |v_{i-1}|_4).$$

Since the imbedding of V to $L_4(\Omega)$ is continuous, $\{u_{i,s}\}$ is bounded in $L_4(\Omega)$. Thus

$$|u_{i,r} - u_{i,s}|_2 \leq C|\mu_{i,r-1} - \mu_{i,s-1}|_4.$$

Boundedness of $\{u_{i,k}\}$ in $V \subset C^{0,\overline{\delta}}(\Omega)$ implies $u_{i,k} \to w$ in $C^{0,\delta}(\overline{\Omega})$ where $0 < \delta < \overline{\delta}$ and $w \in V \cap C^{0,\delta}(\overline{\Omega})$. Hence $u_{i,k} \rightharpoonup w$ in V, and we conclude $\mu_i \equiv \mu, u_i \equiv w$ satisfy (2.8). Moreover taking the limit $k \to \infty$ in (2.9') we have

$$\mu_i \equiv \mu \leq \frac{b_R(v_{i-1} + \alpha(u_i - v_{i-1})) - b_R(v_{i-1})}{u_i - v_{i-1}}$$

which implies (2.9). Thus the proof is complete. □

Remark 5.1. The smoothness $b \in C^2$ has been used in the convergence of the scheme (2.5). For the convergence of the approximation scheme (2.8)–(2.10) it is sufficient to assume $b \in C^1$.

6. Error estimate of the approximation scheme (2.8)–(2.10).

Proof of Theorem 2.6. Let u be a weak solution of problem (2.2) and $\overline{u}^{(n)}$ be a sequence defined in (2.11). We use the ideas from [MNV] to obtain the estimate of the rate of the convergence of $\overline{u}^{(n)}$ to u. Let us integrate the identities (2.4) and (2.8) over $(0, t)$, $t_j \leq t < t_{j+1}$ where j is fixed and subtract them. We obtain

$$\left(b_R(\overline{v}^{(n)}(t))\left(\frac{t - t_j}{\tau}\right) + b_R(\overline{v}^{(n)}(t - \tau))\left(1 - \frac{t - t_j}{\tau}\right) - b_R(u(t)), \varphi\right)$$
$$+ \left(A\int_0^t (\overline{u}^{(n)}(s) - u(s))_x ds, \varphi_x\right) + \left(B\int_0^t \overline{u}^{(n)}(s) - u(s)ds, \varphi_x\right)$$
$$- \left(D\int_0^t \overline{u}^{(n)}(s) - u(s)ds, \varphi\right) = 0.$$

Let us consider $\varphi = \overline{u}^{(n)}(t) - u(t)$ and integrate it over $(0, t)$. We obtain

$$\int_0^t \left(b_R(\overline{v}^{(n)}(s)) \left(\frac{s - t_j}{\tau} \right) + b_R(\overline{v}^{(n)}(s - \tau)) \left(1 - \frac{s - t_j}{\tau} \right) - b_R(u(s)), \overline{u}^{(n)}(s) - u(s) \right) ds$$
$$+ \int_0^t \left(A \int_0^s (\overline{u}^{(n)}(z) - u(z))_x dz, (\overline{u}^{(n)}(s) - u(s))_x \right) ds + \int_0^t \left(B \int_0^s \overline{u}^{(n)}(z) - u(z) dz, \right.$$
$$\left. (\overline{u}^{(n)}(s) - u(s))_x \right) ds - \int_0^t \left(D \int_0^s \overline{u}^{(n)}(z) - u(z) dz, \overline{u}^{(n)}(s) - u(s) \right) ds = 0. \tag{6.1}$$

Let us write it in the form $I_1 + I_2 + I_3 - I_4 = 0$ and estimate the terms. Using Young's inequality, (2.3), (5.5), and (5.6) we have

$$I_1 \geq \frac{\gamma}{2} \int_0^t |\overline{u}^{(n)}(s) - u(s)|_2^2 ds - C\tau. \tag{6.2}$$

We use the relation $\int_0^t (A \int_0^s H(z) dz, H(s)) ds = \frac{1}{2}(A \int_0^t H(s) ds, \int_0^t H(s) ds)$ with $H(s) = (\overline{u}^{(n)}(s) - u(s))_x$ to estimate the second term

$$I_2 \geq \frac{q}{2} \left| \int_0^t (\overline{u}^{(n)}(s) - u(s))_x ds \right|_2^2 . \tag{6.3}$$

Using integration *per partes* in the x variable together with periodicity we obtain

$$I_3 \leq \varepsilon \int_0^t |\overline{u}^{(n)}(s) - u(s)|_2^2 ds + C(\varepsilon) \int_0^t \left| \int_0^s \overline{u}^{(n)}(z) - u(z) dz \right|_2^2 ds$$
$$+ C(\varepsilon) \int_0^t \left| \int_0^s (\overline{u}^{(n)}(z) - u(z))_x dz \right|_2^2 ds. \tag{6.4}$$

$$I_4 \leq \varepsilon \int_0^t |\overline{u}^{(n)}(s) - u(s)|_2^2 ds + C(\varepsilon) \int_0^t \left| \int_0^s \overline{u}^{(n)}(z) - u(z) dz \right|_2^2 ds. \tag{6.5}$$

Using (6.2)–(6.5) in (6.1) we obtain

$$\int_0^t |\overline{u}^{(n)}(s) - u(s)|_2^2 ds + \left| \int_0^t (\overline{u}^{(n)}(s) - u(s))_x ds \right|_2^2 \leq C(\tau + \tau^{2d})$$
$$+ C \int_0^t \left| \int_0^s \overline{u}^{(n)}(z) - u(z) dz \right|_2^2 ds + C \int_0^t \left| \int_0^s (\overline{u}^{(n)}(z) - u(z))_x dz \right|_2^2 ds.$$

Now using Gronwall's lemma we conclude the proof. □

Remark 6.1. Similar convergence results and error estimates (as in §§5 and 6) can be obtained also for the full discretization scheme corresponding to (2.8)–(2.10). In the full discretization, the corresponding elliptic equation is projected on a finite-dimensional subspace $V_\lambda \subset V$; i.e., we look for $u_i^\lambda \in V_\lambda$ such that (2.8) is satisfied $\forall \varphi \in V_\lambda$. We have u^α in the place of $u^{(n)}$ now, with $\alpha = (\tau, \lambda)$ and $V_\lambda \to V$ for $\lambda \to 0$ (in canonical sense). We can proceed in the same way as in [JK1], [JK2], [Ha], and the error estimate

$$|u^\alpha - u|_{L_2(I, L_2)} \leq C \left(\tau + |u_0^\lambda - u_0|_2^2 + \left(\int_I |u - w^\alpha|^2 \right)^{\frac{1}{2}} + \int_I \|u - w^\alpha\|^2 \right) \forall w^\alpha \in L_2(I, V_\lambda)$$

can be obtained, provided $u_0^\lambda \in V_\lambda$, $u_0^\lambda \to u_0$ in $L_2(\Omega)$.

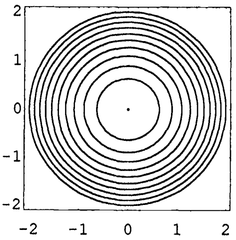


FIG. 1. *Shrinking circle.*

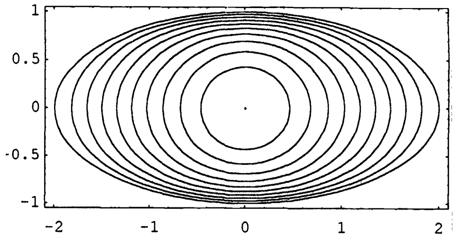


FIG. 2. *Shrinking of the ellipse to the round point.*

7. Discussion on numerical computations. This section is devoted to the presentation of numerical results obtained by approximation schemes (2.5) and (2.8)–(2.10) solving problem (1.1) in the case of anisotropic curve shortening (1.9). All results (graphically documented below) were computed by the semidiscretization (2.8)–(2.10). In each time step we used a few iterations (2.8')–(2.9') to find a couple u_i, μ_i satisfying the convergence condition (2.9). The linear convection–diffusion equation in (2.8') is solved numerically using the so-called "power-law scheme" given in [P]. Then the results were checked using the approximation scheme (2.5); in both procedures we obtained approximately the same degree of accuracy. Some first numerical results computed by approximation scheme (2.8)–(2.10) were presented in [KMi]. We refer to [D] for some other successful numerical method solving (isotropic) curve shortening flow.

First we give a comparison between the known exact results and those obtained numerically; their agreement indicates the usefulness of the method also in cases in which not even qualitative behavior is known.

Simple closed curves. In the isotropic case (1.3) of the curve shortening, a special solution is the circle shrinking in selfsimilar form to the center. Its radius $R(t)$ is then given by $R(t) = \sqrt{2(T-t)}$, where T is the duration of the shrinking. T is also the blow-up time of the corresponding curvature function $k(\theta, t) = 1/\sqrt{2(T-t)}$, which solves the curve shortening equation (1.4) and blows up in the whole interval $\Omega = [0, 2\pi]$. Figure 1 shows the evolution of the circle with $R(0) = 2$ computed by our approximation scheme. The precise blow-up time is 2.0 while the numerically calculated one (i.e., the time when the numerical curvature function is of order 10^6) is 2.02 for $\tau = 0.01$ and 2.007 time units for $\tau = 0.001$. The numerical solution is plotted each 0.2 time units up to time 1.8. In this scale, the exact and numerical solutions are precisely the same. The point in the center represents the solution in time 2.007 and it is a circle with radius about 10^{-6}.

Gage and Hamilton [GH] proved that in the isotropic case, any initial plane convex curve asymptotically behaves as a shrinking circle. Figure 2 illustrates this phenomenon. The initial ellipse gradually reaches a circular shape and finally shrinks to its center. Initially it has axes $a = 2, b = 1$ so the initial area $S(0) = 2\pi$. Because of the relation $S_t = -2\pi$ (see [GH]) we have $T = \frac{S(0)}{2\pi}$. So in our experiment $T = 1$. Numerical blow up is obtained at 1.06 for $\tau = 0.01$ and at 1.005 for $\tau = 0.001$, respectively.

The conservation of closedness of the evolving curves is another important criterion for the approximation scheme precision. In plotting time instants, the curves have been constructed from the computed curvature function $k(\theta, t)$ by (1.7); the integral in (1.7) is evaluated numerically. Because the curves are only given by (1.7) uniquely up to a translation, for determining the real evolution we need to know the motion of at least one point of the curve. For this purpose we use (1.2). Our numerical function $k(\theta, t)$ satisfies the closedness condition very precisely. Thus, on no figure the curve is split. The closedness conservation

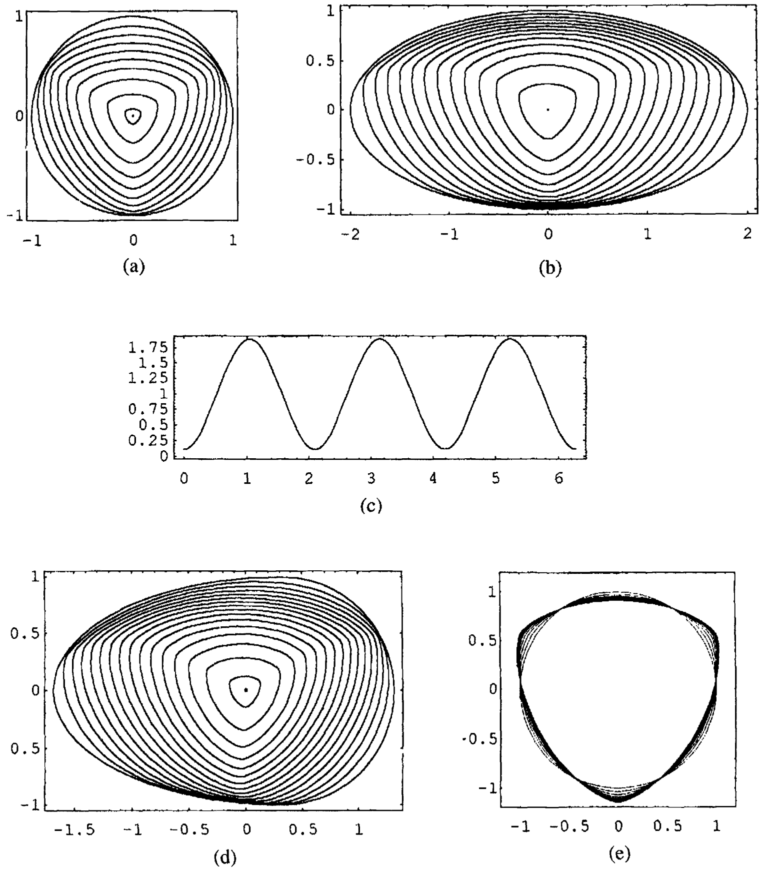


FIG. 3. (a) *Shrinking circle.* (b) *Convergence to the "oval triangle."* (c) *Coefficient* $A(x)$. (d) *Convergence to the "oval triangle."* (e) *Rescaled dynamics.*

is an intrinsic property of the schemes, so we don't have to use any supplementary means to achieve it.

Now, let us consider the anisotropic case. Owing to the data f, β, g, F describing conditions on the free boundary, evolution is more variable. Figure 3a shows the anisotropic shrinking of the circle. In this case $f(\theta) = 1 + \frac{1}{9}\cos 3\theta$, $g(\theta) = 1 - \frac{8}{9}\cos 3\theta$, $\beta(\theta) = 1$, and $F \equiv 0$. The coefficients of the general model (1.1) A, B, D, G are obtained from (1.9). Our numerical solution indicates that a certain "oval triangle" takes over the role of the shrinking circle as the asymptotical shape. In Figure 3c the function $A(x)$ in the interval $[0, 2\pi]$ is plotted. In the model, $A(x)$ represents the diffusion and shrinking terms. The evolution equation (1.2) indicates why the moving of the curve is much slower in the points with θ equal to $0, \frac{2\pi}{3}, \frac{4\pi}{3}$, where A is small, in contrast to the situation in points with θ equal to $\frac{\pi}{3}, \pi, \frac{5\pi}{3}$. Figures 3b and 3d show other initial curves's convergence to the "oval triangle" under the same data f, β, g, F. The asymptotic behavior of the shrinking curves in the anisotropic curve shortening problem has been investigated theoretically now [M]. If $F \equiv 0$ in the anisotropic case, we have for the blow-up time the relation $T = S(0)/\int_0^{2\pi} \frac{g(\theta)}{\beta(\theta)} d\theta$ (see [AG2]). For experiments from Figures 3a, 3b, and 3d the exact T is equal to 0.5, 1.0, 0.75, respectively. By approximation scheme with $\tau = 0.001$ we obtained numerical solutions

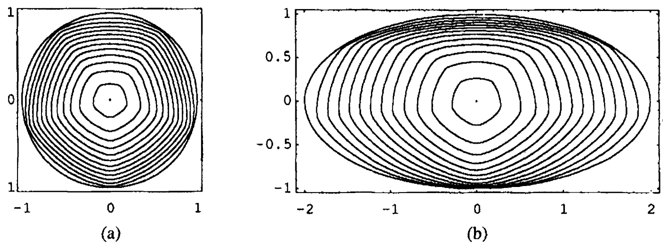


FIG. 4. (a) *Shrinking circle.* (b) *Convergence to the "oval pentagon."*

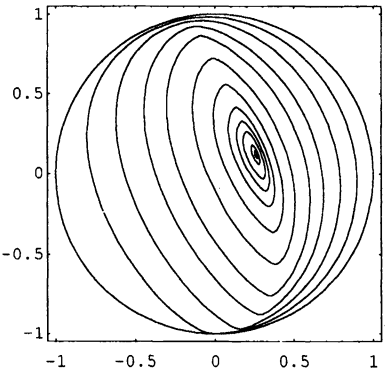


FIG. 5. *General situation of the anisotropic c.s.f.*

with numerical blow-up times 0.508, 1.007, 0.7485, respectively. For another set of data f, β, g, F, when $A(x) = 1 - \frac{5}{6}\cos 5x$, $B(x) = A'(x)$, $D(x) = A(x)$, $G \equiv 0$ the numerical evolution of the initial circle and ellipse is plotted in Figures 4a and 4b, respectively. The numerical calculations indicate that "oval pentagon" is the asymptotical shape for these coefficients. In Figure 5, the coefficients are chosen so as to illustrate the general situation of the anisotropic curve shortening flow; $f(\theta) = 1 + 0.32\cos 2\theta$, $g(\theta) = 1 - 0.96\cos 2\theta$, $\beta(\theta) = 1 + 0.5\cos(\theta - \frac{\pi}{2}) + 0.45\cos(2\theta - \frac{\pi}{2})$, $F \equiv -0.1$. This evolution seems to be a very realistic approximation of the motion of the phase interface.

Let us return to the question of the existence of the asymptotical shape for anisotropically evolving curves. Consider the special case ($F \equiv 0$), when the reference phase cannot increase in time. (For the outer evolution the asymptotical behavior is known; see [AG1], [AG2].) A circle, which is the asymptotical shape for the inner isotropic evolution, minimizes the so-called *isoperimetric ratio* $\mathrm{Iso} = L^2/4\pi S$, where L is the length of the closed curve and S is the enclosed area. For all closed plain curves we have $\mathrm{Iso} \geq 1$ and equality holds only for a circle. For a function $z = z(\theta)$, periodic in $[0, 2\pi]$, we define

$$L(z) = \int_0^{2\pi} \frac{1}{z(\theta)k(\theta)} d\theta, \tag{7.1}$$

$$S(z) = \frac{1}{2}\int_0^{2\pi} \left(\frac{\sin(\theta)}{z(\theta)k(\theta)} \int_0^{\theta} \frac{\cos(\xi)}{z(\xi)k(\xi)} d\xi - \frac{\cos(\theta)}{z(\theta)k(\theta)} \int_0^{\theta} \frac{\sin(\xi)}{z(\xi)k(\xi)} d\xi \right) d\theta \tag{7.2}$$

and call these numbers the *length* of the curve and the *area* it encompasses *with respect to the function* z, respectively. All of our numerical experiments indicate that the *isoperimetric ratio* *with respect to the function* z or the *"anisotropic isoperimetric ratio"*

$$\text{Iso}(z) = \frac{L^2(z)}{4\pi S(z)}, \tag{7.3}$$

where $z(\theta) = \frac{g(\theta)}{\beta(\theta)}$, decreases during curve evolution and converges to 1 (due to numerics approximately). The following numbers concern the same numerical experiment as plotted in Figure 3a. In this case the blow-up time $T = 0.5$. Here we give numerical values of Iso(z) for $z(\theta) = 1 - \frac{8}{9}\cos 3\theta$ in several time moments t given as subscript:

$\text{Iso}_0(z) = 1.114$, $\text{Iso}_{0.02}(z) = 1.093$, $\text{Iso}_{0.04}(z) = 1.070$, $\text{Iso}_{0.06}(z) = 1.047$, $\text{Iso}_{0.08}(z) = 1.028$, $\text{Iso}_{0.10}(z) = 1.014$, $\text{Iso}_{0.12}(z) = 1.0071$, $\text{Iso}_{0.14}(z) = 1.0031$, $\text{Iso}_{0.16}(z) = 1.0011$, $\text{Iso}_{0.36}(z) = 1.0009$, $\text{Iso}_{0.40}(z) = 1.0009$, $\text{Iso}_{0.44}(z) = 1.0009$, $\text{Iso}_{0.48}(z) = 1.0009$.

The ratio Iso(z) decreases and, from a certain time on, there is practically no change. This indicates that the shape is shrinking to a point in a selfsimilar fashion. The question arises as to whether the "oval triangle" minimizes the functional Iso(z) the way the circle minimizes Iso(1).

Let us look at the problem of the asymptotical behavior from another point of view. We consider the special case of g, β, in which $\int_0^{2\pi} \frac{g(\theta)}{\beta(\theta)} d\theta$ is equal to 2π and $F \equiv 0$. Comparing the relations for blow-up time we see that the asymptotics of the shape are different from those in the isotropic case but the growth of the maximal curvature $k_{\max}(t) = |k(\cdot, t)|_\infty$ has to be the same. So, $k_{\max}$ may not grow faster than $1/\sqrt{2(T-t)}$. Let the function $k(\theta, t)$ be the solution of (1.8). Transform the time axis by $\tau = -\frac{1}{2}\log\left(\frac{T-t}{T}\right)$ and define the *rescaled curvature* $K(\theta, \tau) = k(\theta, t)\sqrt{2(T-t)}$. Denote $z(\theta) = \frac{g(\theta)}{\beta(\theta)}$. Since

$$\begin{aligned} K_\tau = \frac{\partial K}{\partial t}\frac{\partial t}{\partial \tau} &= 2(T-t)\left(-\frac{1}{\sqrt{2T-2t}}k + \sqrt{2T-2t}\,k_t\right) \\ &= 2(T-t)\left(-\frac{1}{2T-2t}K + \sqrt{2T-2t}(k^2(zk)_{\theta\theta} + k^2(zk))\right) \\ &= 2(T-t)\left(-\frac{1}{2T-2t}K + \frac{1}{2T-2t}(K^2(zK)_{\theta\theta} + K^2(zK))\right), \end{aligned}$$

the function $K(\theta, \tau)$ is the solution of the *transformed equation*

$$\partial_\tau\left(-\frac{1}{K}\right) = (zK)_{\theta\theta} + zK - \frac{1}{K}. \tag{7.4}$$

If there exists an asymptotical shape of inner evolution with the speed $1/\sqrt{2(T-t)}$ for the given z, then the rescaled curvature of this shape is an equilibrium of equation (7.4). The following numerical experiment illustrates the previous considerations. The initial unit circle from Figure 3a remains the same after rescaling its curvature; now starting the numerical solution of the transformed equation with the same $z(\theta) = 1 - \frac{8}{9}\cos 3\theta$ we obtain the results plotted in Figure 3e. The initial circle changes to "oval triangle" form, where the changes are slower and slower so that we have finally an accentuated steady-state shape whose curvature is the equilibrium of (7.4). The agreement of this steady-state "oval triangle" and the asymptotical shape from Figure 3a may be seen from the evolution of numerical Iso(z) in the rescaled dynamics:

$\text{Iso}_0(z) = 1.114$, $\text{Iso}_{0.04}(z) = 1.072$, $\text{Iso}_{0.08}(z) = 1.033$, $\text{Iso}_{0.12}(z) = 1.011$, $\text{Iso}_{0.16}(z) = 1.003$, $\text{Iso}_{0.20}(z) = 1.0008$, $\text{Iso}_{0.50}(z) = 1.0008$, $\text{Iso}_{0.60}(z) = 1.0009$, $\text{Iso}_{0.70}(z) = 1.0009$, $\text{Iso}_{0.80}(z) = 1.0010$, $\text{Iso}_{0.90}(z) = 1.0010$, $\text{Iso}_{1.0}(z) = 1.0010$.

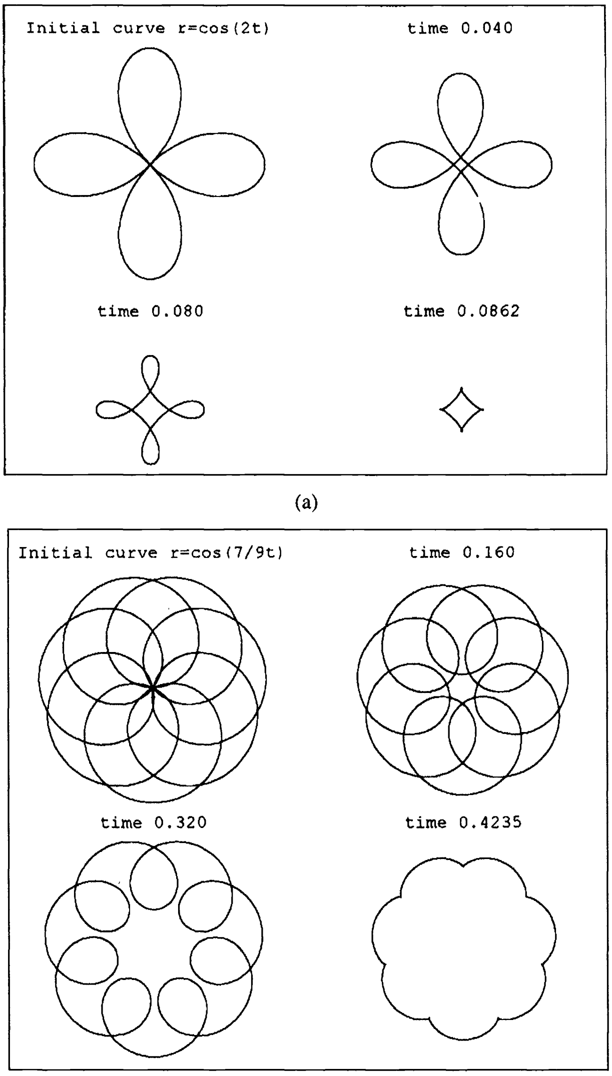

Initial curve r=cos(2t)
time 0.040
time 0.080
time 0.0862
(a)
Initial curve r=cos(7/9t)
time 0.160
time 0.320
time 0.4235
(b)

FIG. 6.

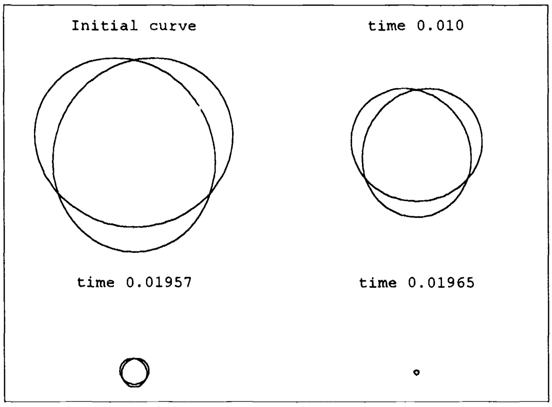
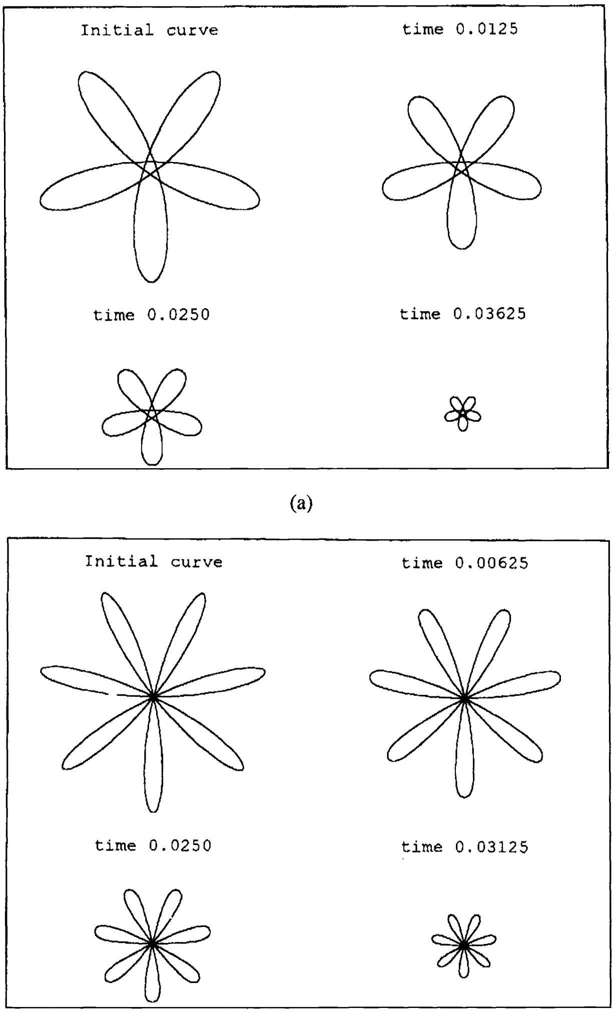

Initial curve
time 0.010
time 0.01957
time 0.01965

FIG. 7.

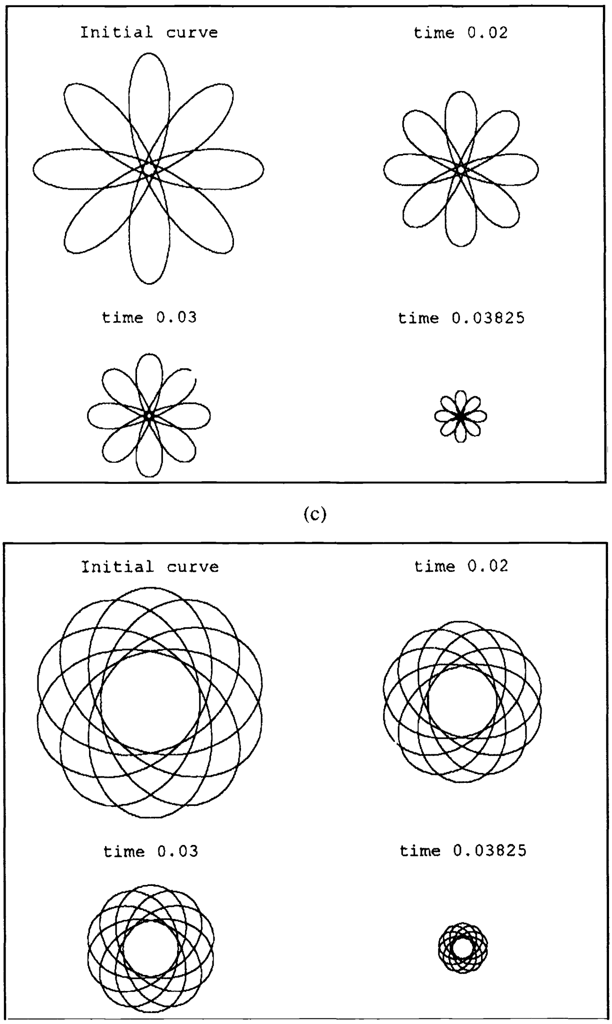


(a)

(b)

FIG. 8.

Closed plain immersed curves. If the index ν of the initial curve is greater than 1 then qualitatively new types of behavior are possible. First we consider the isotropic case. If the initial curve has self-intersections, then the corresponding solution can become singular without shrinking to a point. Figure 6a illustrates the case in which the initial "loops" contract faster than the whole curve and become a so-called *formation of the singularities* in the curve shortening flow. This behavior was studied mainly by Angenent in [A1]. Figure 6b represents this process for another immersed curve with greater inside curvature until the time when the singular curve is formed. The singularities in the form of "hairs" correspond to a blow up of the curvature function $k(\theta, t)$ in several subintervals of Ω. (Every one of these intervals has measure greater than π.) In Figures 6a and b and also in Figures 7–9, we have plotted the initial curve, three states of evolution reconstructed from the numerical solution, the last of which is very near to a blow-up time. As may be seen from previous figures, the ratio of the maximal and minimal curvatures is large and growing quickly in time. In spite of this, our approximation schemes lead to precise results, as can be checked by conservation of curve closedness also very near to the blow up.



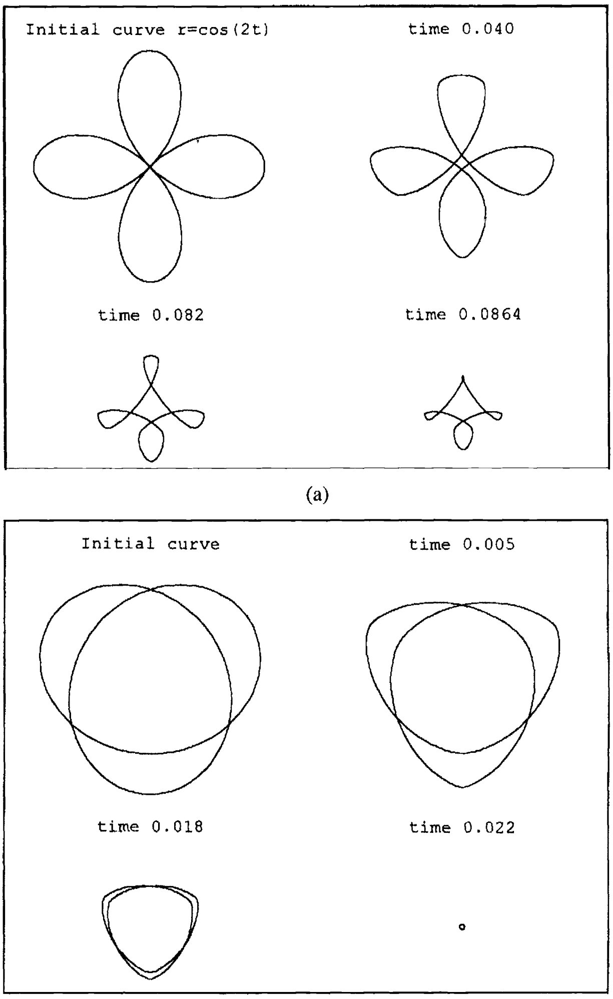
(c)

(d)

FIG. 8. *(cont.)*

Also, in the case in which $\nu > 1$, behavior similar to a shrinking of the simple closed curves into a point is possible. It can happen only in cases in which the maximal curvature does not grow faster than $1/\sqrt{2(T-t)}$. The special initial curve with $\nu = 2$ from Figure 7 gradually winds itself about the circle with index 2 before shrinking to a point. The corresponding curvature function has the blow up in the whole of Ω.

A part of the boundary between the previous two dynamics is formed by the curves which shrink to a point in a selfsimilar manner. This behavior was studied in [ALa] and [EW]. For initial curve $\mathbf{r}$, denote by $n \in \mathbb{N}$ the number of periods of its curvature function $k(\theta)$ in Ω. For curves from Figures 6a–8d it means number of loops. Abresch and Langer [ALa] have shown that, for every integer couple ν, n satisfying $1/2 < \nu/n < \sqrt{2}/2$, there exists one curve which is the homothetic solution of the isotropic curve shortening flow problem. If we add to them the circles with all indexes $\nu \in \mathbb{N}$ we obtain all the so-called *Abresch–Langer functions*. They are steady states of the rescaled equation (7.4) with $z \equiv 1$. Dynamics of



(a)

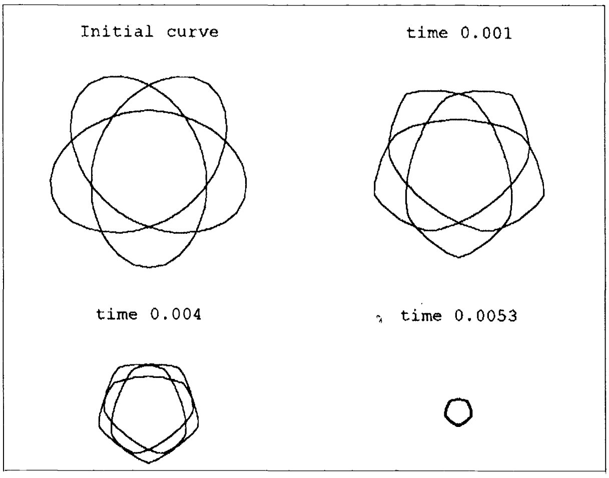
(b)

FIG. 9.

several "almost homothetic solutions" computed by our approximation scheme are plotted in Figures 8a–d.

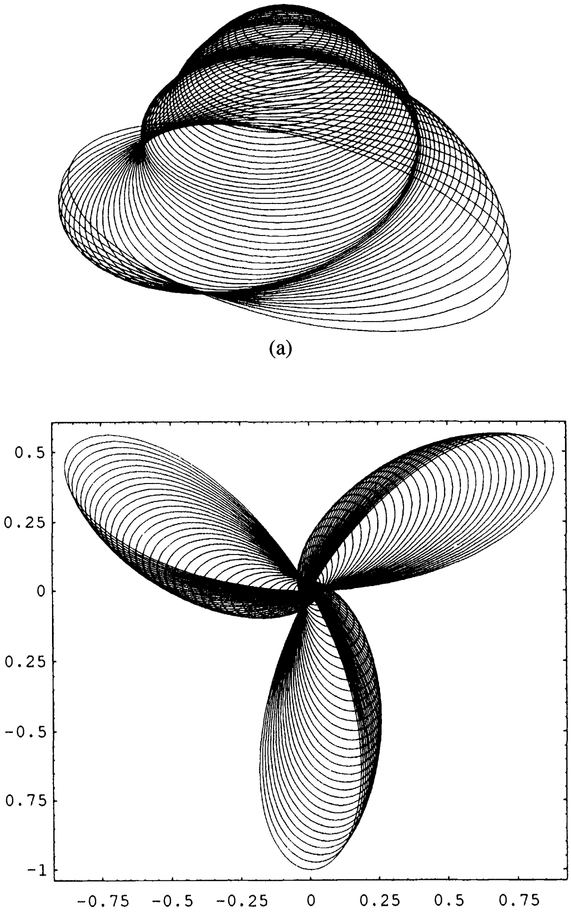
Figures 9a–c concern the anisotropic curve shortening of immersed curves. In Figures 9a–b the evolution is governed by the set of data known from Figures 3a–e, when the asymptotical behavior was the "oval triangle." The curves of Figures 9a–b exhibit a similar tendency. In the first experiment from Figure 9a, the curve becomes singular and one "hair" singularity is eventually formed. The second initial curve from Figure 9b gradually winds itself about this "oval triangle" with index 2 and then shrinks to a point. Winding about the "oval pentagon" with index 3 is shown in Figure 9c. (The coefficients of the model are the same as in experiments documented in Figures 4a and b.) Equilibriums of equation (7.4) with $z \neq 1$ constitute certain generalizations of the Abresch–Langer functions.

Shrinking with supplementary convection term. Let us consider problem (1.1) in the case of (1.9) with one exception, namely $B(x) = (g/\beta)_x + B_1(x)$ for $B_1(x)$ bounded



(c)

FIG. 9. *(cont.)*



(a)

(b)

FIG. 10. *Shrinking circle.*

measurable function periodic in interval $[0, 2\pi\nu]$. Then equation (1.1) can be written in the form (writing k instead of v)

$$\partial_t\left(-\frac{1}{k}\right) = \left(\frac{gk-F}{\beta}\right)_{xx} + \left(\frac{gk-F}{\beta}\right) + (B_1k)_x. \tag{7.5}$$

Equation (7.5) is a certain generalization of the anisotropic curve shortening where we have a supplementary convection term. Our theoretical results guarantee the local existence of a solution and also convergence of approximations. So, such problems can be handled numerically. A question arises whether the found solution satisfies the closedness condition and hence whether it could represent in a sense the evolving phase interface. Several curves which do not split during such "shrinking and whirling around the curve" are presented in Figures 10a and b. As seen in these last figures, if $B_1(x)$ is a positive constant then the curve rotates counterclockwise.

REFERENCES

[ALa] U. Abresch and J. Langer, *The normalized curve shortening flow and homothetic solutions*, J. Differential Geom., 23 (1986), pp. 175–196.

[ALu] H. W. Alt and S. Luckhaus, *Quasilinear elliptic-parabolic differential equations*, Math. Z., 183 (1983), pp. 311–341.

[A1] S. B. Angenent, *On the formation of singularities in the curve shortening flow*, J. Differential Geom., 33 (1991), pp. 601–633.

[A2] ———, *Parabolic equations for curves on surfaces (parts* I *&* II), Ann. of Math., 133 (1991), pp. 171–215.

[A3] ———, *private correspondence*, 1992.

[AG1] S. B. Angenent and M. E. Gurtin, *Multiphase thermomechanics with an interfacial structure* 2. *Evolution of an isothermal interface*, Arch. Rational Mech. Anal., 108 (1989), pp. 323–391.

[AG2] ———, *Anisotropic motion of a phase interface. Well-posedness of the initial value problem and qualitative properties of the interface*, J. Reine Angew. Math., 446 (1994), pp. 1–47.

[D] G. Dziuk, *Convergence of a semi discrete scheme for the curve shortening flow*, Math. Methods Appl. Sci., to appear.

[EW] C. Epstein and M. Weinstein, *A stable manifold theorem for the curve shortening equation*, Comm. Pure Appl. Math., 40 (1987), pp. 119–139.

[FM] A. Friedman and B. McLeod, *Blow-up of solutions of nonlinear degenerate parabolic equations*, Arch. Rational Mech. Anal., 96 (1986), pp. 55–80.

[GH] M. Gage and R. S. Hamilton, *The heat equation shrinking convex plane curves*, J. Differential Geom., 23 (1986), pp. 285–314.

[G1] M. E. Gurtin, *On the two-phase Stefan problem with interfacial energy and entropy*, Arch. Rational Mech. Anal., 96 (1986), pp. 199–241.

[G2] ———, *Multiphase thermomechanics with interfacial structure. Toward a nonequilibrium thermomechanics of two phase materials and entropy*, Arch. Rational Mech. Anal., 100 (1988), pp. 275–312.

[G3] ———, *Multiphase thermomechanics with interfacial structure* 1. *Heat conduction and the capillary balance law*, Arch. Rational Mech. Anal., 104 (1988), pp. 195–221.

[Gr] M. Grayson, *The heat equation shrinks embedded plane curves to round points*, J. Differential Geom., 26 (1987), pp. 285–314.

[Ha] A. Handlovičová, *Error estimates of a linear approximation scheme for nonlinear diffusion problems*, AMUC, 61 (1992), pp. 27–39.

[JK1] W. Jäger and J. Kačur, *Solution of porous medium type systems by linear approximation schemes*, Numer. Math., 60 (1991), pp. 407–427.

[JK2] ———, *Approximation of Degenerate Elliptic-Parabolic Problems by Nondegenerate Elliptic and Parabolic Problems*, Preprint, University of Heidelberg (1991).

[K1] J. Kačur, *Method of Rothe in Evolution Equations*, Teubner-Texte zur Mathematik 80, Teubner, Leipzig, 1985.

[K2] ———, *On a solution of degenerate elliptic-parabolic systems in Orlicz-Sobolev spaces*. I, II, Math. Z., 203 (1990), pp. 569–579.

[K3] ———, *On a solution of a generalized system of von Kármán equations*, Apl. Mat., 26 (1981), pp. 437–448.

[KHK] J. Kačur, A. Handlovičová, and M. Kačurová, *Solution of nonlinear diffusion problems by linear approximation schemes*, SIAM J. Numer. Anal., 30 (1993), pp. 1703–1722.

[KMi] J. Kačur and K. Mikula, *Numerical solution of the anisotropic curve shortening flow*, in Proc. International Symposium on Numerical Analysis, ISNA'92, Prague, pp. 108–121.

[KJF] A. Kufner, O. John, and S. Fučík, *Function Spaces*, Academia, Prague, 1977.

[L] O. A. Ladyženskaja, *Linear and Quasilinear Elliptic Equations*, Nauka, Moskva, 1964.

[LSU] O. A. Ladyženskaja, V. A. Solonnikov, and N. N. Uraceva, *Linear and quasilinear equations of parabolic type*, in Translation of Mathematical Monographs 23, American Mathematical Society, Providence, RI, 1968.

[M] H. Matano, *private communication*, EQUADIFF 8 in Bratislava, 1993.

[MNV] E. Magenes, R. H. Nochetto, and C. Verdi, *Energy error estimates for a linear scheme to approximate nonlinear parabolic problems*, Math. Mod. Numer. Anal., 21 (1987), pp. 655–678.

[P] S. Patankar, *Numerical Heat Transfer and Fluid Flow*, Hemisphere Publishing Corp., New York, 1980.

[W] P. A. Watterson, *Force-Free Magnetic Evolution in the Reversed-Field Pinch*, thesis, Cambridge University, 1985.

[Z] T. Zimmer, *Theorie und Numerik einer degenerierten, parabolischen Differentialgleichung*, Preprint, Bayreuth University, (1991).

SIAM J. SCI. COMPUT.
Vol. 17, No. 6, pp. 1328–1346, November 1996

OSCILLATION ABSORPTION FINITE ELEMENT METHODS FOR CONVECTION–DIFFUSION PROBLEMS*

W. LAYTON[†] AND B. POLMAN[‡]

Abstract. This paper proposes a new approach to eliminating overshoots and undershoots when using finite element methods (FEM) for convection-dominated convection–diffusion problems. The first scheme involves an additional nonlinear "absorption" term which frequently improves the solution quality of both the usual Galerkin FEM and the streamline diffusion FEM. Mathematical support for the method is given in the form of global estimates for the overshoot, undershoot, and the basic error in the approximate solution. Although nonlinear, the additional term is in fact *monotone*. Another approach is to strongly impose zero overshoot and undershoot via a variational inequality formulation. Finally we consider the possibility of variational inequality postprocessing to remove overshoots and undershoots.

Key words. convection–diffusion, nonlinear absorption, monotone operators

AMS subject classifications. 65N12, 65N30

1. Introduction.

This paper considers the finite element approximation of convection–diffusion problems, such as

$$\text{(1.1)} \qquad \begin{cases} -\epsilon\Delta u + a_1u_x + a_2u_y + gu = f(x,y) & \text{for } (x,y)\in\Omega\subset\mathbb{R}^2, \\ u(x,y)=\alpha(x,y) & \text{for } (x,y)\in\partial\Omega. \end{cases}$$

In (1.1) $\vec{a} = (a_1, a_2)$ represents a known convection field transporting a substance whose concentration is given by $u(x, y)$, ϵ is a (small) diffusion coefficient, and $g(x, y)$ is a linear absorption term. The method studied involves adding to a standard Galerkin finite element method (FEM) or streamline diffusion FEM an additional nonlinear absorption term $(A(u^h), v)$. This term has the effect of eliminating overshoots and undershoots in the concentration $u(x, y)$. These overshoots and undershoots are well known to cause difficulties when (1.1) is coupled with, for example, chemical reaction terms.

There exist well-known flux-splitting finite difference methods for 1-dimensional conservation laws which do not introduce nonphysical maxima. These are typically extended to 2-dimensions via operator splitting along coordinate directions and applying the 1-dimensional methods in each coordinate direction; see, e.g., [4, 17] for representative examples. Finite element research has taken a slightly different tack, focusing on arbitrary order methods applicable to complex geometries and unstructured meshes. This approach has given rise to the streamline diffusion method of [8]. The success of both approaches for problems closely related has inspired attempts at obtaining a synthesis of the best features of both, e.g., [9, 15]. One possible synthesis of the two methods is to solve the linear system iteratively (or, equivalently, by imbedding it in an evolutionary problem) and, at each step, eliminating out-of-range answers as they occur. This is exactly equivalent to solving a variational inequality approximation by a projective iterative method; see §3.

To present the new method, and a first example, let $\Pi^h(\Omega)$ be an edge-to-edge triangulation of Ω with maximum triangle diameter h. $X^h(\Omega)$ will denote a typical Lagrange-type finite element space associated with $\Pi^h(\Omega)$. In our examples we shall take conforming C^0-piecewise linears for $\Pi^h(\Omega)$, any degree being possible in principle. The combinations of nonlinear

*Received by the editors November 30, 1993; accepted for publication (in revised form) May 30, 1995.

[†] Department of Mathematics, University of Nijmegen, 6525 ED Nijmegen, The Netherlands. Permanent address: Mathematics Department, University of Pittsburgh, Pittsburgh, PA (wjl@vms.cis.pitt.edu). This author was partially supported by N.W.O. Grant B61-222.

[‡] Department of Mathematics, University of Nijmegen, 6525 ED Nijmegen, The Netherlands (polman@sci.kun.nl).

absorption plus both the usual Galerkin and the streamline diffusion FEMs are studied herein, as well as two related ideas. For example, the usual Galerkin FEM plus artificial absorption computes $u^h \in X^h(\Omega)$ via the following: for all $v \in X^h(\Omega) \cap H^1(\Omega)$,

$$(1.2) \qquad \begin{cases} (\epsilon \nabla u^h, \nabla v) + (\vec{a} \cdot \nabla u^h + g u^h, v) + (A(u^h), v) = (f, v), \\ u^h(N_j) = \alpha(N_j) \quad \text{for all nodes } N_j \text{ on } \partial\Omega. \end{cases}$$

The term $(A(u^h), v)$, given by (1.5) below, incorporates into (1.2) the fact that global bounds of the form $u_{\min} \le u(x, y) \le u_{\max}$ are known for the true solution, and the approximate solution should satisfy these bounds as well. Consider the following first (favorable) example.

Example 1.1. The following problem was solved on a uniform rectilinear mesh, each square divided into two triangles, using conforming linears with (a) standard Galerkin FEM and (b) standard Galerkin FEM with the oscillation absorption term added, method (1.2). The parameter values were $h = \frac{1}{25}$, $\epsilon = 10^{-3}$, $\Omega = (0, 1) \times (0, 1)$, $a_1(x, y) = c(1 - cx)$, $a_2(x, y) = s(1 - sy)$, where $c = \cos\theta$, $s = \sin\theta$, and $\theta = 210°$ (this is the test problem in [2]). Consider

$$(1.3) \qquad \begin{cases} -\epsilon \Delta u + a_1(x, y) u_x + a_2(x, y) u_y = 0 & \text{for } (x, y) \in \Omega, \\ u(x, y) = \alpha(x, y) & \text{for } (x, y) \in \partial\Omega, \end{cases}$$

where $\alpha(x, y) = 1$, if $12y - 5x \ge .3$, $\alpha(x, y) = 0$ otherwise. The maximum principle immediately gives $0 \le u(x, y) \le 1$ for the true solution to (1.3). This information is incorporated into the absorption term via

$$A(u)(x, y) := \rho^{-1} \left[\min\{u(x, y), 0\} + \max\{u(x, y) - 1, 0\}\right],$$

where ρ was chosen (see §§2 and 4) to be $\rho = h^2$.

The difference between Figures 1a and 1b is enormous. In §4 a comparison is also given with the streamline diffusion FEM and the streamline diffusion FEM with the $(A(u^h), v)$ term added.

The basic principle of (1.2) is that L^∞-information from the continuous problem can be incorporated in a *consistent*, *highly accurate* manner into (1.2) via the additional $(A(u^h), v)$ term. Suppose, for example, that the maximum principle for (1.1) gives the bounds

$$(1.4) \qquad u_{\min} \le u(x, y) \le u_{\max}, \qquad (x, y) \in \bar{\Omega}$$

for the true solution to (1.1). Then $A(u)$ is given by

$$(1.5) \qquad A(u)(x, y) := \rho^{-1} \left[\min\{u(x, y) - u_{\min}, 0\} + \max\{u(x, y) - u_{\max}, 0\}\right],$$

where ρ is a "small" parameter. There are many applications where bounds of the form (1.4) are immediate, for example natural convection, see [3], possibly including convection–diffusion of a pollutant as well.

Note that $A(u)$ is a penalty-type term. In §2 mathematical support for the addition of the $(A(u), v)$ term to both the usual Galerkin and the streamline diffusion formulations of (1.1) is given. Note that (1.5) is the usual penalty term employed in approximating *variational inequality* problems. Thus, associated with (1.2) is another method based upon imposing the constraint (1.4) strongly via a variational inequality formulation. Based upon the fact that the true solution lies in the relevant constraint set, optimal order estimates are given for the variational inequality approach in §3.

Additionally, there are at least two attractive generalizations of (1.2), (1.4), (1.5) involving further subdivisions of, respectively, the range of u and the domain Ω. These two will now be briefly described. A more detailed treatment is beyond the scope of this report.

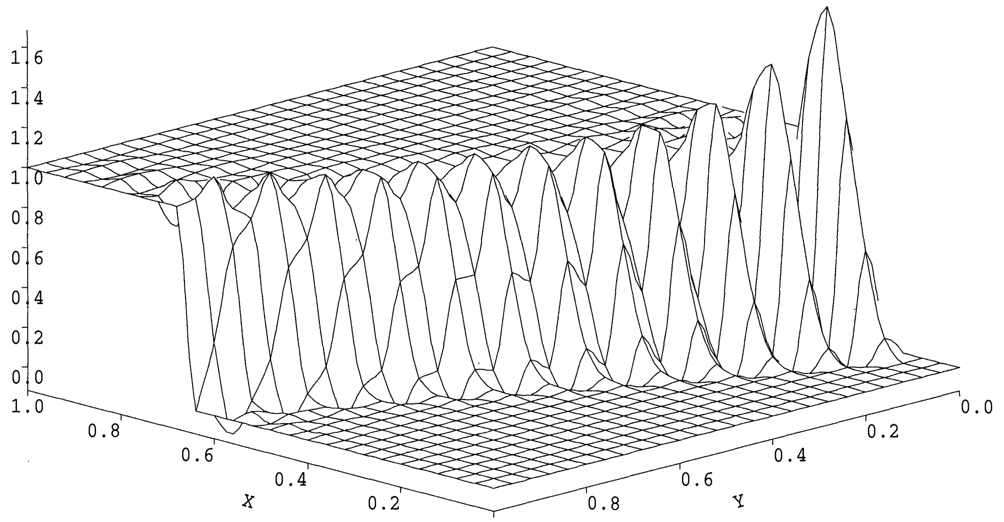


FIG. 1a. *Usual Galerkin FEM approximation to problem* (1.3).

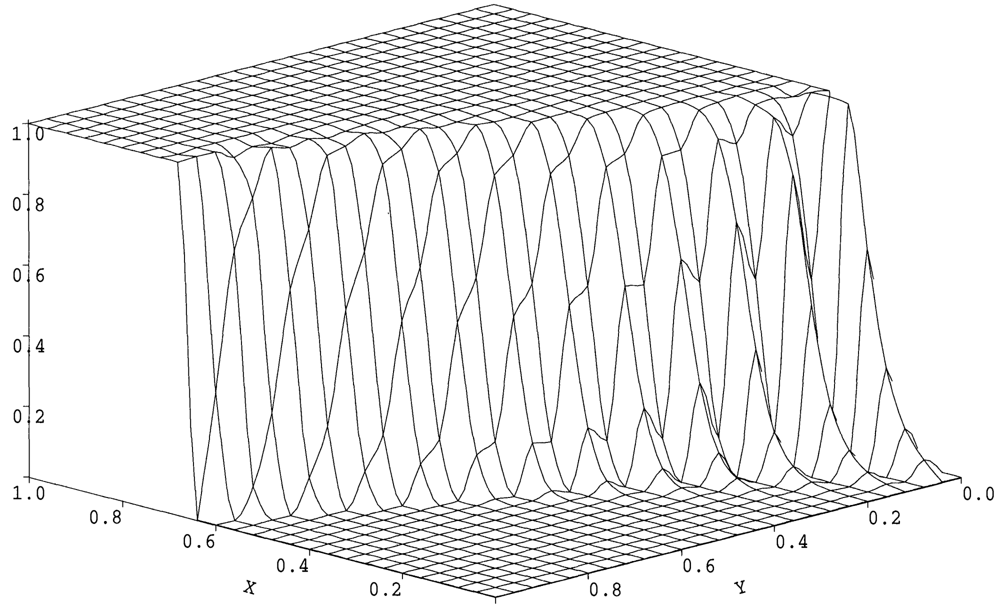


FIG. 1b. *Oscillation absorption FEM approximation* (1.2) *to problem* (1.3).

Generalization 1: Range decompositions. Consider the problem with no internal sources: seek $u(x, y)$ satisfying

$$-\epsilon\Delta u + au_x + bu_y = 0 \quad \text{in } \Omega, \qquad u = g \quad \text{on } \partial\Omega,$$

where, for compactness, $0 \leq g(x, y) \leq 1$ (and hence $0 \leq u(x, y) \leq 1$) in $\bar{\Omega}$. For a positive integer J decompose the range of u by $0 = U_0 < U_1 < \cdots < U_J = 1$. This induces an additive decomposition of the boundary data

$$g(x, y) = \sum_{j=1}^{J} g_j(x, y) \quad \text{with } g_j \in C^0(\partial\Omega)$$

and $U_0 \leq g_j(x, y) \leq (U_j - U_{j-1})$ by

$$U_{j-1} + g_j(x, y) = \begin{cases} U_{j-1} & \text{if } g(x, y) \leq U_{j-1}, \\ g(x, y) & \text{if } U_{j-1} \leq g(x, y) \leq U_j, \\ U_j & \text{if } g(x, y) \geq U_j. \end{cases}$$

A corresponding additive decomposition of the solution is induced, $u(x, y) = \sum_{j=1}^{J} u_j(x, y)$, where $u_j(x, y)$ satisfies

$$(1.6) \qquad -\epsilon\Delta u_j + a u_{j,x} + b u_{j,y} = 0 \quad \text{in } \Omega, \quad u_j = g_j \quad \text{on } \partial\Omega,$$

so that

$$(1.7) \qquad 0 \leq u_j(x, y) \leq (U_j - U_{j-1}).$$

The J uncoupled problems (1.6) can now be solved subject to the constraint (1.7), either strongly or weakly imposed as in §§2 and 3 to give u_j^h. Thereupon the global approximate solution is reassembled as $u^h = \sum_{j=1}^{J} u_j^h$. In this way, oscillations around certain critical solution values (the U_j's) can be eliminated.

Generalization 2: Further division of Ω. If Ω_j denotes possibly overlapping subdomains of Ω, $\Omega = \bigcup_{j=1,J} \Omega_j$, the term $A(u)$ could be modified to read

$$\begin{aligned} A(u)(x, y) := \rho^{-1}[&\min\{u(x, y) - \min\{u(x, y) \mid (x, y) \in \partial\Omega_j\}, 0\} \\ &+ \max\{u(x, y) - \max\{u(x, y) \mid (x, y) \in \partial\Omega_j\}, 0\}] \quad \text{for } (x, y) \in \Omega_j. \end{aligned} \tag{1.8}$$

In the limiting case where for each node N_j, $\Omega_j = \Omega(N_j)$ is the union of the triangles containing N_j, (1.8) resembles a "flux limiter," well known in finite difference theory.

The analysis herein is, however, restricted to the problems $\Omega_j \equiv \Omega$ (§2) and $\Omega_j \equiv \Omega$ and $\rho \to 0$ (§3).

Numerical experiments are given in §4. Provided one has estimates of $u_{\max}$ and $u_{\min}$, improvement in solution quality is consistently seen. Sometimes the improvement is quite dramatic and sometimes it is only moderate. In all cases, overshoots and undershoots are eliminated. In some cases, however, oscillations between the correct maximum and minimum values persist when using the usual (centered) Galerkin FEM. In these first experiments, damped Newton methods were used to solve the nonlinear penalized problem (without spectacular efficiency for the Galerkin FEM). Since the nonlinear term makes the system "more" monotone, there are other attractive and highly parallel solution procedures which are to be recommended; see [12, 13].

2. Mathematical support for the nonlinear absorption term. For brevity in the presentation, consider the problem with homogeneous, boundary conditions:

$$(2.1) \qquad \begin{cases} L_\epsilon u \equiv -\epsilon\Delta u + \vec{a} \cdot \nabla u + gu = f & \text{in } \Omega \text{ (polygonal domain in } \mathbb{R}^d), \\ u = 0 & \text{on } \partial\Omega, \\ g - \frac{1}{2}\text{div}(\vec{a})^t \geq g_{\min} \geq 0 & \text{in } \Omega. \end{cases}$$

We suppose á priori bounds on the true solution to (2.1) are known and given by (1.4):

$$(2.2) \qquad (-\infty \leq) u_{\min} \leq u(x, y) \leq u_{\max} (\leq +\infty), \qquad (x, y) \in \bar{\Omega}.$$

Let $X = \mathrm{H}_0^1(\Omega)$ and X^h denote a conforming finite element space, then $X^h \subset X$, where h represents a typical maximum element diameter. The method is now given as follows. Suppose that the true solution of (2.1) satisfies, for all $v \in X$,

$$B(u, v) = F(v), \tag{2.3}$$

where $B(u, \cdot)$ is linear on X and $F(\cdot)$ is a bounded linear functional on X.

Choose a small parameter $\rho > 0$ and define, for $w \in X$,

$$A(w)(\underline{x}) := \rho^{-1}\left[(w(\underline{x}) - u_{\min})_- + (w(\underline{x}) - u_{\max})_+\right], \tag{2.4}$$

where we use the notation $v_- = \min\{v, 0\}$ and $v_+ = \max\{v, 0\}$. We consider an approximation of the following form. For $B^h(\cdot, \cdot) : X \times X \mapsto \mathbb{R}$ and $F^h(\cdot) : X \mapsto \mathbb{R}$ continuous bilinear and linear, respectively,

$$\left\{\begin{array}{l} \text{seek } u^h \in X^h \text{ satisfying} \\ B^h(u^h, v) + (A(u^h), v) = F^h(v) \quad \forall v \in X^h. \end{array}\right. \tag{2.5}$$

We shall suppose that there are norms $\|\cdot\|_a, \|\cdot\|_b$ on X equivalent to the H^1-norm, such that

$$\left\{\begin{array}{ll} B^h(w, w) \geq C_1\|w\|_a^2 & \forall w \in X, \\ B^h(w, v) \leq C_2\|w\|_a\|v\|_b & \forall v, w \in X, \end{array}\right. \tag{2.6}$$

where C_1, C_2 are constants which as we shall see later can be chosen independent of h and ϵ in our applications. The question of existence of a solution to the nonlinear problem (2.5) is resolved in the following proposition.

PROPOSITION 2.1. *Assume* (2.6) *holds. Let $T^h : X^h \mapsto X^h$ be defined by*

$$(T^h(w), v) := B^h(w, v) + (A(w), v) \quad \forall v \in X^h. \tag{2.7}$$

Then T^h is well defined and strictly monotone for all $v, w \in X^h$,

$$(T^h(v) - T^h(w), v - w) \geq C_1\|v - w\|_a^2 + \frac{1}{\rho}\|v - w\|_{L^2(\Omega_1)}^2,$$

where $\Omega_1 := \{\underline{x} \in \Omega \mid [v(\underline{x}) < u_{\min}$ and $w(\underline{x}) < u_{\min}]$ or $[v(\underline{x}) > u_{\max}$ and $w(\underline{x}) > u_{\max}]\}$. Similarly, we can let $T : X \mapsto X$ be defined by

$$(T(w), v) := B^h(w, v) + (A(w), v) \quad \forall v \in X.$$

If (2.2) *holds, then for all $v \in X$,*

$$(T(v) - T(u), v - u) \geq C_1\|v - u\|_a^2 + \frac{1}{\rho}\left\{\|\min\{v - u_{\min}, 0\}\|^2 + \|\max\{v - u_{\max}, 0\}\|^2\right\}.$$

Proof. First note that as (2.7) is linear and continuous in v, T^h is well defined by the Riesz representation theorem. Further the $A(\cdot)$ term in (2.7) satisfies

$$(A(v) - A(w), v - w) \geq \frac{1}{\rho}\|v - w\|_{L^2(\Omega_1)}^2 \quad \forall v, w \in X$$

(note that a simple calculation shows that we only get the L^2-norm over Ω_1) and for all $v \in X$ (note that $A(u) = 0$),

$$(A(v) - A(u), v - u) \geq \frac{1}{\rho}\left\{\|\min\{v - u_{\min}, 0\}\|^2 + \|\max\{v - u_{\max}, 0\}\|^2\right\}.$$

Combining these two inequalities with coercivity of $B^h(\cdot,\cdot)$ demonstrates the claimed monotonicity. □

COROLLARY 2.1. *Supposing* (2.6) *holds,* $u^h \in X^h$*, the solution of* (2.5)*, exists uniquely.*

Proof. This proof follows immediately from monotonicity of T^h and classical results of Minty [14]. □

Since the operator T is monotone, standard techniques can be used to obtain an error bound. Adapting these appropriately yields the next theorem.

THEOREM 2.1. *Suppose* (2.6) *and* (2.2) *hold. Then there exists a constant C independent of* ρ *and h such that*

$$\begin{aligned}\|u-u^h\|_a^2 + \frac{1}{\rho}\|(u^h-u_{\min})_-\|^2 + \frac{1}{\rho}\|(u^h-u_{\max})_+\|^2 \\ \le C \inf_{\chi\in X^h}\left\{\|u-\chi\|_a^2 + \|u-\chi\|_b^2 + \frac{1}{\rho}\|u-\chi\|_{L^2(\Omega)}^2\right\} \\ + C\left\{\sup_{w^h\in X^h}\frac{|[B^h-B](u,w^h)|}{\|w^h\|_a}\right\}^2 \\ + C\left\{\sup_{w^h\in X^h}\frac{|[F^h-F](w^h)|}{\|w^h\|_a}\right\}^2.\end{aligned} \tag{2.8}$$

Proof. Let $e = u^h - u$ and for $\chi \in X^h$, $\eta = \chi - u$ and $\phi = u^h - \chi$. Then from Proposition 2.1,

$$\begin{aligned}C_1\|e\|_a^2 + \frac{1}{\rho}\|(u^h-u_{\min})_-\|^2 + \frac{1}{\rho}\|(u^h-u_{\max})_+\|^2 \\ \le (T(u^h)-T(u), u^h-u) \\ = (T(u^h)-T(u), u^h-\chi) + (T(u^h)-T(u), \chi-u).\end{aligned} \tag{2.9}$$

Consider the first term of the right-hand side (RHS) as $A(u)\equiv 0$ and u^h satisfies (2.5):

$$\begin{aligned}(T(u^h)-T(u),\phi) &= B^h(e,\phi) + (A(u^h),\phi) \\ &= B^h(u^h,\phi) - B^h(u,\phi) + F^h(\phi) - B^h(u^h,\phi) + [B(u,\phi)-F(u)] \\ &= [B-B^h](u,\phi) + [F^h-F](\phi) \\ &\le \left(\sup_{w^h\in X^h}\frac{[B^h-B](u,w^h)}{\|w^h\|_a}\right)\|\phi\|_a + \left(\sup_{w^h\in X^h}\frac{[F^h-F](w^h)}{\|w^h\|_a}\right)\|\phi\|_a \\ &\le [C_B + C_F]\,(\|u-\chi\|_a + \|e\|_a),\end{aligned} \tag{2.10}$$

where $C_B := \sup_{w^h\in X^h}\frac{[B^h-B](u,w^h)}{\|w^h\|_a}$ and $C_F := \sup_{w^h\in X^h}\frac{[F^h-F](w^h)}{\|w^h\|_a}$.

Now consider the second term on the RHS of (2.9):

$$\begin{aligned}(T(u^h)-T(u),\chi-u) &= B^h(e,\eta) + (A(u^h),\eta) \\ &\le C_2\|e\|_a\|\eta\|_b + \|A(u^h)\|\|\eta\| \\ &\le C_2\|e\|_a\|\eta\|_b + \frac{1}{\rho}\Big(\|(u^h-u_{\min})_-\| + \|(u^h-u_{\max})_+\|\Big)\|\eta\|\end{aligned}$$

$$\le C_2\|e\|_a\|\eta\|_b + \frac{1}{4\rho}\Big(\|(u^h - u_{\min})_-\| + \|(u^h - u_{\max})_+\|\Big)^2 + \frac{1}{\rho}\|\eta\|^2$$

$$\le C_2\|e\|_a\|\eta\|_b + \frac{1}{2\rho}\Big(\|(u^h - u_{\min})_-\|^2 + \|(u^h - u_{\max})_+\|^2\Big) + \frac{1}{\rho}\|\eta\|^2. \tag{2.11}$$

Inserting (2.10) and (2.11) into (2.9) yields

$$C_1\|e\|_a^2 + \frac{1}{2\rho}\Big(\|(u^h - u_{\min})_-\|^2 + \|(u^h - u_{\max})_+\|^2\Big)$$
$$\le C_2\|e\|_a\|\eta\|_b + [C_B + C_F](\|\eta\|_a + \|e\|_a) + \frac{1}{\rho}\|\eta\|^2.$$

Hiding the $\|e\|_a$ terms in the left-hand side (LHS) and taking the infimum over $\chi \in X^h$ yields (2.8). □

As an application of Theorem 2.1, we give the error estimates which (2.8) yields when applied to the usual Galerkin method and the streamline diffusion method by using C^0 conforming linear elements.

COROLLARY 2.2. *Suppose* $B^h(v, w) = B(v, w) = \int_\Omega \epsilon\nabla v\nabla w + (\vec{a}\nabla v + gv)w dx\, g_{\min} > 0$, *and* $F^h(w) = F(w) = \int_\Omega f w dx$. *Then* $u - u^h$ *satisfies*

$$\epsilon\|\nabla(u - u^h)\|^2 + \|u - u^h\|^2 + \frac{1}{\rho}\|(u^h - u_{\min})_-\|^2 + \frac{1}{\rho}\|(u^h - u_{\max})_+\|^2$$
$$\le C \inf_{\chi\in X^h}\left\{2\epsilon\|\nabla(u - \chi)\|^2 + (2 + \epsilon^{-1} + \rho^{-1})\|u - \chi\|^2\right\}. \tag{2.12}$$

Proof. Set $\|w\|_a^2 := \epsilon\|\nabla w\|^2 + \|w\|^2$ and $\|w\|_b^2 := \epsilon\|\nabla w\|^2 + (1 + \epsilon^{-1})\|w\|^2$. Then (2.6) is fulfilled with C_1 and C_2 independent of ϵ and h and the result follows directly from Theorem 2.1. □

For the streamline diffusion method a weighting parameter $\delta = O(h)$ is selected. For a detailed discussion of streamline diffusion methods, see [2, 8, 10]. The presentation of this method is a bit simpler when using for X^h the space of C^0 conforming piecewise linears. The main idea is to embed (2.1) into

$$-\delta\nabla\cdot(\vec{a}L_\epsilon u) + L_\epsilon u = -\delta\nabla\cdot(\vec{a}f) + f \tag{2.13}$$

which is consistent with the original problem if u is sufficiently smooth ($u \in H^2(\Omega)\cap \mathrm{H}_0^1(\Omega)$). This leads to the following variational formulation:

$$B_\delta(u, v) = F_\delta(v) \qquad \forall v \in \mathrm{H}_0^1(\Omega), \text{ for } u \in H^2 \cap \mathrm{H}_0^1(\Omega),$$

where

$$B_\delta(u, v) := \int_\Omega \epsilon\nabla u\nabla v - \epsilon\delta(\Delta u)\vec{a}\cdot\nabla v + (\vec{a}\cdot\nabla u + gu)(v + \delta\vec{a}\cdot\nabla v)dx, \tag{2.14a}$$

$$F_\delta(v) := \int_\Omega f(v + \delta\vec{a}\cdot\nabla v)dx. \tag{2.14b}$$

Using linear basisfunctions this results in

$$B_\delta^h(u, v) = F_\delta^h(v) \qquad \forall v \in X^h,$$

where

$$B_\delta^h(w^h, v^h) := \int_\Omega \epsilon \nabla w^h \nabla v^h + (\vec{a} \cdot \nabla w^h + g w^h)(v^h + \delta \vec{a} \cdot \nabla v^h) dx, \tag{2.15a}$$

and

$$F_\delta^h(v^h) := F_\delta(v^h). \tag{2.15b}$$

Note that in B_δ^h we have dropped the term $\epsilon\delta(\Delta u)\vec{a} \cdot \nabla v$. This is due to the fact that if we write the integral over Ω as a sum of integrals over the elements, then in each element $\Delta v \equiv 0$, where v is a linear basisfunction.

COROLLARY 2.3. *Let X^h be the space of C^0 conforming piecewise linear basisfunctions and $\delta = O(h)$, where h is the maximum triangle diameter. Let B_δ, B_δ^h, F_δ, and F_δ^h as in* (2.14), (2.15). *Then*

$$B_\delta(u, v) = B_\delta^h(u, v) - \epsilon\delta \int_\Omega (\Delta u)\vec{a} \cdot \nabla v dx$$

and the error in method (2.5) *satisfies*

$$\begin{aligned} &\epsilon \|\nabla(u - u^h)\|^2 + \delta \|\vec{a} \cdot \nabla(u - u^h)\|^2 + \|u - u^h\|^2 + \frac{1}{\rho}\|(u^h - u_{\min})_-\|^2 + \frac{1}{\rho}\|(u^h - u_{\max})_+\|^2 \\ &\leq C \inf_{\chi \in X^h} \left\{2\epsilon \|\nabla(u - \chi)\|^2 + 2\delta \|\vec{a} \cdot \nabla(u - \chi)\|^2 + (2 + \delta^{-1} + \rho^{-1})\|u - \chi\|^2\right\}. \end{aligned} \tag{2.16}$$

Proof. Set

$$\|w\|_a^2 := \epsilon \|\nabla w\|^2 + \delta \|\vec{a} \cdot \nabla w\|^2 + \|w\|^2,$$
$$\|w\|_b^2 := \epsilon \|\nabla w\|^2 + \delta \|\vec{a} \cdot \nabla w\|^2 + (1 + \delta^{-1})\|w\|^2.$$

Again (2.6) is fulfilled and the result follows by Theorem 2.1. □

Formally balancing terms on the RHS of (2.12) and (2.16) suggests the following scaling for the penalty parameter ρ.

Usual Galerkin FEM + penalty term.

$\epsilon = O(1) \Rightarrow \rho = O(h^2)$,

$\epsilon = O(h) \Rightarrow \rho = O(h)$,

$\epsilon \leq O(h^2) \Rightarrow \rho = O(\epsilon)$.

Streamline diffusion FEM + penalty term.

$\rho = O(\delta) = O(h)$.

In our first experiments, however, solution quality did not seem very sensitive to the actual choice of ρ, unlike the convergence of the damped Newton solution procedure employed (only in the case of the usual Galerkin method).

3. The variational inequality approach. For compactness of exposition, assume that the boundary conditions are homogeneous and that one knows á priori a closed, convex set $\mathbb{K}$ in $\mathrm{H}_0^1(\Omega)$ in which the true solution lies. Therefore, consider

$$\begin{cases} -\epsilon\Delta u + \vec{a} \cdot \nabla u + g u = f(x, y) & \text{in } \Omega \subset \mathbb{R}^d, \\ u = 0 & \text{on } \partial\Omega. \end{cases} \tag{3.1}$$

Assume $g - \frac{1}{2}\mathrm{div}(\vec{a})^t \geq g_{\min} \geq 0$ in $\bar{\Omega}$ and that, á priori, the true solution of (3.1) satisfies

$$u \in \mathbb{K}, \quad \mathbb{K} \subset \mathrm{H}_0^1(\Omega) \text{ a closed, convex set.} \tag{3.2}$$

LEMMA 3.1. *Suppose the true solution u of* (3.1), (3.2) *satisfies the following variational equality: for X a Hilbert space, $u \in X$,*

$$B(u, v) - F(v) = 0 \quad \forall v \in X, \tag{3.3}$$

where $B(u, v)$ is a continuous and coercive bilinear form,

$$|B(u, v)| \leq C_2(\epsilon)\|u\|_X\|v\|_X, \quad |B(w, w)| \geq C_1(\epsilon)\|w\|_X^2 \qquad \forall v, w \in X, \tag{3.4}$$

and $F(v)$ is a bounded linear functional on X. If $\mathbb{K} \subset X$ is a closed convex set in X, then the variational inequality

$$\begin{cases} \text{seek } \tilde{u} \in \mathbb{K} \text{ satisfying} \\ B(\tilde{u}, v - \tilde{u}) - F(v - \tilde{u}) \geq 0 \quad \forall v \in \mathbb{K} \end{cases} \tag{3.5}$$

has a unique solution $\tilde{u}$. If $u \in \mathbb{K}$, then $\tilde{u} \equiv u$.

Proof. Existence and uniqueness of $\tilde{u}$ in (3.5) follows from (3.4) and standard results in variational inequalities; see Kinderlehrer and Stampacchia [11, Chap. 2, Thm. 2.1]. Likewise, existence and uniqueness for (3.3) follows by the Lax–Milgram lemma. As (3.3) and $u \in \mathbb{K}$ imply that u satisfies (3.5), necessarily $u = \tilde{u}$. □

Next, consider the error in approximating the solution of the variational inequality (3.5). Since we wish to include both streamline diffusion and usual Galerkin FEMs, the classic error analysis of Falk [6] and Mosco and Strang [16] must be enlarged a bit to include perturbations of the bilinear forms, nonsymmetric problems, and the dependence upon ϵ in the final estimates. To this end, suppose the true solution $u \in X$ to the problem satisfies the following *variational equality*:

$$B(u, v) = F(v) \quad \forall v \in X, \tag{3.6}$$

where $B(u, \cdot) : X \mapsto \mathbb{R}$ is linear and $F : X \mapsto \mathbb{R}$ is a linear functional. Likewise, let $X^h \subset X$ be a closed subspace and define

$$\mathbb{K}^h := X^h \cap \mathbb{K}, \qquad \mathbb{K} \text{ a closed, convex set in } X. \tag{3.7}$$

We assume $\mathbb{K}^h \neq \emptyset$ and $u \in \mathbb{K}$. The approximate solution we consider is calculated via the following variational *inequality*:

$$\begin{cases} \text{find } u^h \in \mathbb{K}^h \text{ satisfying} \\ B^h(u^h, v^h - u^h) - F^h(v^h - u^h) \geq 0 \quad \forall v^h \in \mathbb{K}^h, \end{cases} \tag{3.8}$$

where $B^h : X^h \times X^h \mapsto \mathbb{R}$, $F^h : X^h \mapsto \mathbb{R}$ are, respectively, bilinear and linear functionals approximating $B(\cdot, \cdot)$ and $F(\cdot)$. $B^h(\cdot.\cdot)$ is assumed to satisfy the following.

Assumption 3.1. There are norms $\|\cdot\|_a$, $\|\cdot\|_b$ on X such that for positive constants $C_{1,2}$, independent of h and ϵ,

$$B^h(w, w) \geq C_1\|w\|_a^2 \qquad \forall w \in X,$$
$$B^h(v, w) \leq C_2\|v\|_a\|w\|_b \qquad \forall v, w \in X.$$

THEOREM 3.1. *Suppose Assumption* 3.1 *holds. Then, $u^h \in \mathbb{K}^h$ satisfying* (3.8) *exists and is unique. Further, if $u \in \mathbb{K}$, then the error $u - u^h$ satisfies*

$$\|u - u^h\|_a \leq C_2C_1^{-1} \inf_{\chi \in \mathbb{K}^h} \|u - \chi\|_b + 2C_1^{-1} \sup_{w^h \in X^h} \frac{[F - F^h](w^h)}{\|w_h\|_a}$$
$$+ 2C_1^{-1} \sup_{w^h \in X^h} \frac{[B^h - B](u, w^h)}{\|w_h\|_a}.$$

Proof. Letting $\chi \in \mathbb{K}^h$ be arbitrary, expand the error inequality as follows:

$$\begin{aligned} C_1\|u-u^h\|_a^2 &\le B^h(u-u^h, u-u^h) \\ &= B^h(u-u^h, u-\chi) - B^h(u^h, \chi-u^h) + B^h(u, \chi-u^h) \\ &\le B^h(u-u^h, u-\chi) + F(\chi - u^h) - F^h(\chi - u^h) + [B^h - B](u, \chi-u^h) \end{aligned}$$

using (3.8), (3.6)

$$\le C_2\|u-u^h\|_a\|u-\chi\|_b + \sup_{w^h\in X^h}\frac{[F-F^h](w^h)}{\|w_h\|_a}\|\chi-u^h\|_a + \sup_{w^h\in X^h}\frac{[B^h-B](u,w^h)}{\|w_h\|_a}\|\chi-u^h\|_a.$$

Now insert $\|\chi-u^h\|_a \le \|u-u^h\|_a + \|u-\chi\|_a$ and divide by $\|u-u^h\|_a$ in the resulting inequality. Taking the infimum over χ yields

$$C_1\|u-u^h\|_a \le C_2\inf_{\chi\in\mathbb{K}^h}\|u-\chi\|_b + 2\sup_{w^h\in X^h}\frac{[F-F^h](w^h)}{\|w_h\|_a} + 2\sup_{w^h\in X^h}\frac{[B^h-B](u,w^h)}{\|w_h\|_a},$$

which is the claimed error estimate. □

Remark 3.1. The conditions in Assumption 3.1 can be relaxed somewhat. For example, essentially the same result holds if X is replaced by X^h in Assumption 3.1. On the other hand, if coercivity is weakened to the inf-sup condition, a similar error estimate can be given but *existence* of the approximate solution (to (3.8)) is then an open question.

As applications of Theorem 3.1, consider the bilinear forms $B^h(\cdot,\cdot)$ based upon (1) the usual Galerkin FEM and (2) the streamline diffusion FEM by using linear elements.

Example 3.1. *The usual Galerkin FEM.* In the usual Galerkin FEM with exact integration $B^h(\cdot,\cdot) = B(\cdot,\cdot)$, $F^h(\cdot) = F(\cdot)$, $\|w\|_a^2 := \epsilon\|\nabla w\|^2 + \|w\|^2$, and $\|w\|_b^2 := \epsilon\|\nabla w\|^2 + (1+\epsilon^{-1})\|w\|^2$. The constants $C_{1,2}$ are $O(1)$ constants provided $g_{\min} > 0$.

COROLLARY 3.1. *Suppose* $g_{\min} > 0$, *then the error in* (3.8) *when using the usual Galerkin formulation satisfies*

$$\left[\epsilon\|\nabla(u-u^h)\|^2 + \|u-u^h\|^2\right]^{\frac{1}{2}} \le C\inf_{\chi\in\mathbb{K}^h}\left[\epsilon\|\nabla(u-\chi)\|^2 + (1+\epsilon^{-1})\|u-\chi\|^2\right]^{\frac{1}{2}}.$$

Example 3.2. *The streamline diffusion FEM.* For $u \in H^2(\Omega)\cap \mathrm{H}_0^1(\Omega)$, the solution of the continuous problem satisfies the following: for $\delta > 0$, a parameter

$$B_\delta(u,v) = F_\delta(v) \qquad \forall v \in \mathrm{H}_0^1(\Omega), \text{ for } u \in H^2\cap\mathrm{H}_0^1(\Omega),$$

with B_δ and F_δ as defined in (2.14). The streamline diffusion formulation of (3.8) with *linear* elements is defined as follows: seek $u^h \in \mathbb{K}^h$ satisfying

$$B_\delta^h(u^h, v^h-u^h) \ge F_\delta^h(v^h-u^h) \qquad \forall v^h \in \mathbb{K}^h, \tag{3.9}$$

with B_δ^h and F_δ^h as defined in (2.15). With

$$\begin{aligned} \|w\|_a^2 &:= \epsilon\|\nabla w\|^2 + \delta\|\vec{a}\cdot\nabla w\|^2 + \|w\|^2, \\ \|w\|_b^2 &:= \epsilon\|\nabla w\|^2 + \delta\|\vec{a}\cdot\nabla w\|^2 + (1+\delta^{-1})\|w\|^2 \end{aligned}$$

as the choice of norms, B_δ^h satisfies Assumption 3.1 provided $g_{\min} > 0$. Note that $[F - F^h] \equiv 0$ and $[B_\delta - B_\delta^h](u, w^h) = -\epsilon\delta \int_\Omega \Delta u(\vec{a} \cdot \nabla w^h)dx$ which we can bound by the following:

$$\left|[B_\delta - B_\delta^h](u, w^h)\right| \leq \epsilon\delta \|\Delta u\| \|\vec{a} \cdot \nabla w^h\| \leq \epsilon\delta^{\frac{1}{2}} \|\Delta u\| \|w^h\|_a.$$

Using these estimates in Theorem 3.1, we state our next corollary.

COROLLARY 3.2. *Suppose* $g_{\min} > 0$. *Then, method* (3.9) *satisfies*

$$\begin{aligned}\left[\epsilon\|\nabla(u - u^h)\|^2 + \delta\|\vec{a} \cdot \nabla(u - u^h)\|^2 + \|u - u^h\|^2\right]^{\frac{1}{2}} \\ \leq C \inf_{\chi \in \mathbb{K}^h} \left[\epsilon\|\nabla(u - \chi)\|^2 + \delta\|\vec{a} \cdot \nabla(u - \chi)\|^2 \right. \\ \left. + (1 + \delta^{-1})\|u - \chi\|^2\right]^{\frac{1}{2}} + C\epsilon\delta^{\frac{1}{2}}\|\Delta u\|.\end{aligned}$$

Remark 3.2. *Concerning the interpolation error in* $\mathbb{K}^h$. The above global estimates involve terms of the form $\inf_{v^h \in \mathbb{K}^h} \|u - v^h\|_b$. It seems to be an open question to give an asymptotic estimate of this for elements where the usual nodal basisfunctions are not nonnegative. For conforming linears, for example, if $u \in \mathbb{K}$ then $I^h(u)$, the usual nodal interpolant of u, lies in $\mathbb{K}^h$ so the infimum over K^h will yield the same asymptotic error estimates as the infimum over X^h. For more general elements, the estimates that are known, see, e.g., Wong [18], can be summarized in the following lemma.

LEMMA 3.2. *Suppose* X *is a separable Hilbert space,* $\mathbb{K} \subset X$ *is a closed, nonempty, convex subset of* X. *Let* X^h, $0 \leq h \leq 1$, *be a nested* 1*-parameter family of finite-dimensional subspaces which become dense in* X *as* $h \to 0$. *If the interior of* $\mathbb{K}$ *is nonempty, then for* $h \leq h_0$, $K^h := \mathbb{K} \cap X^h$ *is nonempty. If additionally,* $u \in \operatorname{Int}(\mathbb{K})$, *then there is an* h_1, *such that for* $h \leq h_1 (\leq h_0)$,

$$\inf_{V^h \in \mathbb{K}^h} \|u - v^h\|_X \leq C(1 + \|u\|_X) \inf_{w^h \in X^h} \|u - w^h\|_X.$$

Proof. This is proven in Wong [18]. □

Unfortunately, the application of this lemma is highly restricted due to the assumption that Int($\mathbb{K}$) is nonempty. We state the following well-known fact for completeness.

LEMMA 3.3. *Let* Ω *be a domain with smooth boundary and* $\mathbb{K} = \{v \in H^1(\Omega) \mid 0 \leq v \leq 1\}$. *Then, if* $\Omega \subset \mathbb{R}^1$, Int($\mathbb{K}$) *is nonempty. If* $\Omega \subset \mathbb{R}^d$, $d > 1$, Int($\mathbb{K}$) *is empty.*

Proof. The first claim follows from the Sobolev embedding theorem. The second follows since there are functions in $H^1(\Omega)$ for $d > 1$ with $\|\phi\|_1 = 1$ yet ϕ is unbounded. Thus, if $u \in \mathbb{K}$, $(u + \epsilon\phi) \notin \mathbb{K}$ for any $\epsilon > 0$ and Int($\mathbb{K}$) is empty. □

4. Numerical experiments. We present a few numerical examples which illustrate the approaches of the previous sections. The elimination of the overshoots and undershoots is illustrated as well as improvement in the approximate solution between its maximum and minimum values. The test problem considered is the problem in Example 1.1:

$$\begin{cases} -\epsilon\Delta u + a_1(x, y)u_x + a_2(x, y)u_y = 0 & \text{for } (x, y) \in \Omega = [0, 1]^2, \\ u(x, y) = \alpha(x, y) & \text{for } (x, y) \in \partial\Omega, \end{cases} \tag{4.1}$$

where $a_1(x, y) = c(1 - cx)$, $a_2(x, y) = s(1 - sy)$, where $c = \cos\theta$, $s = \sin\theta$ (note that $\underline{\nabla} \cdot \vec{a} = -1$ so that the associated bilinear form is coercive). On the boundary we have $\alpha(x, y) = 1$ if $12y - 5x \geq .3$, $\alpha(x, y) = 0$ otherwise.

The problem is discretized using the standard Galerkin FEM and the streamline diffusion FEM by using the standard linear nodal basisfunctions $\{\phi_i^h\}_{i=1}^N$ on a uniform mesh of isosceles right-angled triangles. This leads to a nonsymmetric linear system of equations (for more details on the actual implementation, see [2])

$$K_h \underline{x}^h = \underline{f}^h \tag{4.2}$$

which we solve by a generalized conjugate gradient method with either a point incomplete (for the Galerkin method) or a block incomplete (for the streamline diffusion method) factorization as preconditioner (see [1] and [2]). To implement our new method, we only have to add the nonlinear absorption term to the system (4.2). To simplify the practical implementation, we replace the terms

$$\int\!\!\int_{\Omega} \left(A\left(\sum_{i=1}^{N} x_i^h \phi_i^h \right), \phi_j^h \right) d\Omega \tag{4.3}$$

which are close to $M_h \tilde{A}(\underline{x}^h)$, where M_h is the finite element mass matrix, $(M_h)_{i,j} = (\phi_i^h, \phi_j^h)$, and $\tilde{A} : \mathbb{R}^N \mapsto \mathbb{R}^N$ is defined by

$$\tilde{A}(\underline{x}^h)_i = \frac{1}{\rho}[\min\{x_i^h, 0\} + \max\{x_i^h - 1, 0\}], \qquad i = 1, 2, \ldots, N, \tag{4.4}$$

by just $h^2 \tilde{A}(\underline{x}^h)$ (the h^2 term stems from the scaling of M_h).

It is easy to verify that this corresponds with a monotone operator so the theory of §2 remains applicable. Adding the term $h^2 \tilde{A}(\underline{x}^h)$ to (4.2) yields the following *nonlinear* nonsymmetric system of equations to be solved:

$$K_h \underline{x}^h + h^2 \tilde{A}(\underline{x}^h) = \underline{f}^h. \tag{4.5}$$

Equivalently, seek a zero of the functional $\mathcal{F} : \mathbb{R}^N \mapsto \mathbb{R}^N$ defined by

$$\mathcal{F}(\underline{x}^h) = K_h \underline{x}^h + h^2 \tilde{A}(\underline{x}^h) - \underline{f}^h. \tag{4.6}$$

In these preliminary tests we solved this by a damped Newton method. For a detailed analysis of this type of method see, for instance, [5]. We like to remark here that we have not tried to find an optimal solution method for this system since our main interest here lies in the actual solution; on the other hand, in most cases the solution procedure was in fact very efficient. The damped Newton method employed takes the following form:

```
Choose $\underline{x}^{(0)}$, $k = 0$
DO
    Solve for $\underline{d}^{(k)}$ : $\left[ K_h + \frac{h^2}{\rho} D_h(\underline{x}^{(k)}) \right] \underline{d}^{(k)} = -\mathcal{F}(\underline{x}^{(k)})$
    $\underline{x}^{(k+1)} = \underline{x}^{(k)} + \tau_k \underline{d}^{(k)}$
    $k = k + 1$
WHILE ( $\|\mathcal{F}(\underline{x}^{(k)})\|_2 > 10^{-4}$ ),
```

where $D_h(\underline{x})$ is a diagonal matrix defined by

$$(D_h(\underline{x}))_{i,i} = \begin{cases} 1 & \text{if } x_i < 0 \text{ or } x_i > 1, \\ 0 & \text{otherwise.} \end{cases}$$

TABLE 1

Number of iterations and L_2- and L_∞-norms of overshoots and undershoots for problem (4.1) *using* $\theta = 60°$, *Galerkin FEM.*

$\epsilon \backslash n$	32		64		128	
$10E-1$	$\rho = 0.977E-2$		$\rho = 0.244E-2$		$\rho = 0.61E-3$	
	0	0	0	0	0	0
	0	0	0	0	0	0
	$it_N = 0$	$it_{CG} = 18$	$it_N = 0$	$it_{CG} = 31$	$it_N = 0$	$it_{CG} = 59$
$10E-3$	$\rho = 0.977E-2$		$\rho = 0.244E-2$		$\rho = 0.61E-3$	
	$0.57708E0$	$0.28984E-2$	$0.31662E0$	$0.37947E-3$	$0.20413E0$	$0.99588E-5$
	$0.62121E-1$	$0.16470E-3$	$0.22207E-1$	$0.10091E-4$	$0.17670E-2$	$0.49091E-6$
	$it_N = 3$	$it_{CG} = 152$	$it_N = 3$	$it_{CG} = 106$	$it_N = 2$	$it_{CG} = 90$
	$0.16225E-1$	$0.11982E-3$	$0.79626E-2$	$0.13623E-4$	$0.39860E-2$	$0.83568E-8$
	$0.15514E-2$	$0.51860E-5$	$0.35024E-3$	$0.23519E-6$	$0.31456E-4$	$0.90988E-10$

TABLE 2

Number of iterations and L_2- and L_∞-norms of overshoots and undershoots for problem (4.1) *using* $\theta = 210°$, *Galerkin FEM.*

$\epsilon \backslash n$	32		64		128	
$10E-1$	$\rho = 0.977E-2$		$\rho = 0.244E-2$		$\rho = 0.61E-3$	
	0	0	0	0	0	0
	0	0	0	0	0	0
	$it_N = 0$	$it_{CG} = 19$	$it_N = 0$	$it_{CG} = 36$	$it_N = 0$	$it_{CG} = 70$
$10E-3$	$\rho = 0.977E-2$		$\rho = 0.244E-2$		$\rho = 0.61E-3$	
	$0.84353E-1$	$0.78402E0$	$0.52135E-1$	$0.64231E0$	$0.13318E-1$	$0.47868E0$
	$0.47284E-2$	$0.42018E-1$	$0.95222E-3$	$0.18157E-1$	$0.12832E-3$	$0.87815E-2$
	$it_N = 7$	$it_{CG} = 15$	$it_N = 6$	$it_{CG} = 11$	$it_N = 3$	$it_{CG} = 10$
	$0.66463E-2$	$0.50833E-1$	$0.28948E-2$	$0.24699E-1$	$0.49450E-3$	$0.10868E-1$
	$0.23096E-3$	$0.21857E-2$	$0.46280E-4$	$0.63066E-3$	$0.40775E-5$	$0.19063E-3$

The damping parameter τ_k is determined by a line search with the objective of decreasing $\|\mathcal{F}(\underline{x})\|_2$. One can easily verify that the computed Newton direction is a descent direction for $\|\mathcal{F}(\underline{x})\|_2$ at $\underline{x}^{(k)}$ so that there exist $\tau_k > 0$ for which $\|\mathcal{F}(\underline{x}^{(k+1)})\|_2 < \|\mathcal{F}(\underline{x}^{(k)})\|_2$. In all cases $\tau_k = 1$ (i.e., a full Newton step) is tried first. If this gives a decrease in $\|\mathcal{F}(\underline{x})\|_2$, it is accepted; otherwise, a backtrack algorithm is used based on quadratic and, if necessary, cubic interpolation.

In the following tables we display the infinity norm and the discrete L_2-norm of the undershoots and overshoots of the solutions of (4.1) and (4.5). Furthermore we give the number of Newton iterations (it_N) and the average number of conjugate gradient iterations per Newton step (it_{CG}). Tables 1 and 2 correspond to the Galerkin FEM formulation with $\theta = 60°$ and $\theta = 210°$ and Tables 3 and 4 correspond to the streamline diffusion formulation. In these tables we have chosen the parameter ρ in accordance with the remark at the end of §2. That is, $\rho = O(h)$ for the streamline diffusion formulation, and $\rho = O(h^2)$ if $\epsilon > h$, $\rho = O(\epsilon)$ otherwise for the Galerkin formulation.

In each of these tables, we display

$$\begin{array}{c|c} \|(\underline{x}^h)_-\|_\infty & \|(\underline{x}^h)_+\|_\infty \\ \hline \|(\underline{x}^h)_-\|_{L_2} & \|(\underline{x}^h)_+\|_{L_2} \end{array}$$

first for the solution of (4.1) and below we give the number of iterations for the solution of (4.5).

TABLE 3

Number of iterations and L_2- and L_∞-norms of overshoots and undershoots for problem (4.1) *using* $\theta = 60°$, *streamline diffusion FEM.*

$\epsilon \backslash n$	32	64	128
$10E-1$	$\rho = 0.3125E-1$ 0 0 0 0 $it_N = 0 \quad it_{CG} = 8$	$\rho = 0.15625E-1$ 0 0 0 0 $it_N = 0 \quad it_{CG} = 14$	$\rho = 0.61E-3$ 0 0 0 0 $it_N = 0 \quad it_{CG} = 25$
$10E-3$	$\rho = 0.3125E-1$ 0.14148E0 0.15803E−2 0.13469E−1 0.68486E−4 $it_N = 1 \quad it_{CG} = 3$ 0.15197E−1 0.40508E−3 0.14937E−2 0.13539E−4	$\rho = 0.15625E-1$ 0.25029E−1 0.95503E−4 0.41447E−3 0.94978E−5 $it_N = 1 \quad it_{CG} = 4$ 0.47302E−2 0.81507E−5 0.73919E−4 0.14106E−6	$\rho = 0.78125E-2$ 0.14215E−2 0.97640E−4 0.11428E−4 0.14359E−4 $it_N = 0 \quad it_{CG} = 5$ 0.14215E−2 0.97640E−4 0.11428E−4 0.14359E−4
$10E-5$	$\rho = 0.3125E-1$ 0.81119E0 0.10192E0 0.48997E−1 0.80981E−2 $it_N = 5 \quad it_{CG} = 6$ 0.39651E−1 0.50531E−2 0.36094E−2 0.26599E−3	$\rho = 0.15625E-1$ 0.83080E0 0.73586E−1 0.34587E−1 0.50323E−2 $it_N = 4 \quad it_{CG} = 4$ 0.36436E−1 0.26733E−2 0.23221E−2 0.10797E−3	$\rho = 0.78125E-2$ 0.82927E0 0.44233E−1 0.24176E−1 0.24589E−2 $it_N = 7 \quad it_{CG} = 3$ 0.36447E−1 0.30074E−2 0.15800E−2 0.42053E−4

TABLE 4

Number of iterations and L_2- and L_∞-norms of overshoots and undershoots for problem (4.1) *using* $\theta = 210°$, *streamline diffusion FEM.*

$\epsilon \backslash n$	32	64	128
$10E-1$	$\rho = 0.3125E-1$ 0 0 0 0 $it_N = 0 \quad it_{CG} = 8$	$\rho = 0.15625E-1$ 0 0 0 0 $it_N = 0 \quad it_{CG} = 15$	$\rho = 0.61E-3$ 0 0 0 0 $it_N = 0 \quad it_{CG} = 30$
$10E-3$	$\rho = 0.3125E-1$ 0.38320E−1 0.55808E−1 0.42470E−2 0.54222E−2 $it_N = 3 \quad it_{CG} = 4$ 0.21302E−1 0.21994E−1 0.93083E−3 0.11901E−2	$\rho = 0.15625E-1$ 0.27544E−1 0.40526E−1 0.11408E−2 0.16390E−2 $it_N = 3 \quad it_{CG} = 4$ 0.15263E−1 0.16281E−1 0.33379E−3 0.43761E−3	$\rho = 0.78125E-2$ 0.84764E−2 0.15242E−1 0.11971E−3 0.28427E−3 $it_N = 1 \quad it_{CG} = 3$ 0.48255E−2 0.63733E−2 0.53039E−4 0.87614E−4
$10E-5$	$\rho = 0.3125E-1$ 0.54452E−1 0.72917E−1 0.99571E−2 0.10813E−1 $it_N = 3 \quad it_{CG} = 5$ 0.27646E−1 0.27797E−1 0.12642E−2 0.15436E−2	$\rho = 0.15625E-1$ 0.53142E−1 0.71934E−1 0.75410E−2 0.80042E−2 $it_N = 4 \quad it_{CG} = 6$ 0.27198E−1 0.27237E−1 0.63356E−3 0.77070E−3	$\rho = 0.78125E-2$ 0.51869E−1 0.71394E−1 0.55305E−2 0.57897E−2 $it_N = 4 \quad it_{CG} = 9$ 0.26845E−1 0.26921E−1 0.31716E−3 0.38528E−3

Note that when $it_N = 0$, it_{CG} is precisely the number of iterations needed to solve problem (4.1). In those cases the diffusion is large enough to avoid overshoots and undershoots. In Tables 1 and 2 we have not included the results for $\epsilon = 10^{-5}$ since in that case the solution of problem (4.1), that is, the standard Galerkin formulation, is extremely oscillatory and the Newton process for the solution of problem (4.5) starting with this initial guess converges very slowly. Furthermore, the preconditioner used for this problem behaves very poorly so every

TABLE 5

Number of iterations and L_2- and L_∞-norms of overshoots and undershoots for problem (4.1) *using* $\theta = 210°$, $\epsilon = 0.001$, *streamline diffusion FEM.*

ρ	it_N	it_{CG}	$\|(\underline{x}^h)_-\|_\infty$	$\|(\underline{x}^h)_+\|_\infty$	$\|(\underline{x}^h)_-\|_{L_2}$	$\|(\underline{x}^h)_+\|_{L_2}$
1	0	4	$0.27544E-1$	$0.40526E-1$	$0.11408E-2$	$0.16390E-2$
$10E-1$	1	5	$0.23641E-1$	$0.32760E-1$	$0.69396E-3$	$0.99995E-3$
$10E-2$	3	5	$0.12397E-1$	$0.12823E-1$	$0.25829E-3$	$0.32700E-3$
$10E-3$	3	4	$0.22140E-2$	$0.22259E-2$	$0.40784E-4$	$0.46861E-4$
$10E-4$	3	4	$0.24107E-3$	$0.24215E-3$	$0.43606E-5$	$0.49368E-5$
$10E-5$	3	4	$0.24324E-4$	$0.24433E-4$	$0.43912E-6$	$0.49617E-6$

iteration in the Newton process becomes very costly. Further investigations for this specific case are needed; here our main interest is in showing the feasibility of our approach.

Finally, in Table 5 we show the norms of overshoots and undershoots for varying ρ for the streamline diffusion FEM for the same model problem with $\theta = 210°$, $n = 64$, and $\epsilon = 1E{-}3$. From this table we clearly see the linear dependence on ρ.

To conclude this section we show in Figures 2a and 2b the typical qualitative improvement when adding the nonlinear absorption term to problem (4.1) in the case of a streamline diffusion approximation. In §2 we already saw an example for the Galerkin FEM approximation.

To show the local nature of the absorption term added, we show in Figure 2c the overshoot and undershoot and in Figure 2d we show the final nonlinear absorption term in the Newton process.

5. Overshoot and undershoot elimination by postprocessing. At this stage it is difficult to attribute the large number of iterations in the case of the Galerkin FEM formulation to anything except the lack of computational experience with highly nonsymmetric variational inequalities. It is possible that the solution of highly nonsymmetric variational inequalities can be circumvented if the basic, linear method produces a moderately accurate approximate solution. In this case the overshoots and undershoots can likely be eliminated by postprocessing via a symmetric variational inequality. Suppose that an approximation $u^h \in X^h(\Omega) \subset X$ is calculated by

$$B(u^h, v) = F(v), \qquad v \in X^h. \tag{5.1}$$

Here (5.1) is a nonsymmetric *linear* system. Now consider postprocessing u^h to produce an approximate solution $\Pi u^h \in \mathbb{K}^h$. The simplest method to program is simply to cut u^h off at $u_{\max}$ and $u_{\min}$ when $u^h > u_{\max}$ and $u^h < u_{\min}$, respectively. Perhaps this is acceptable; however, we study a different postprocessor for which mathematical support can be given and which does not require significantly more programmer time and computational effort than the simple clipping procedure.

Let $a(\cdot,\cdot)$ denote a *symmetric* coercive and continuous bilinear form on X. Then Π^h will denote the projection of X^h onto $\mathbb{K}^h$.

DEFINITION 5.1. *Given* $w^h \in X^h$, $\Pi^h w^h \in \mathbb{K}^h$ *is the unique element of* $\mathbb{K}^h$ *satisfying*

$$\|w^h - \Pi^h w^h\|_A = \inf_{\chi \in \mathbb{K}^h} \|w^h - \chi\|_A,$$

where $\|v\|_A = a(v, v)^{1/2}$ *is the induced norm.*

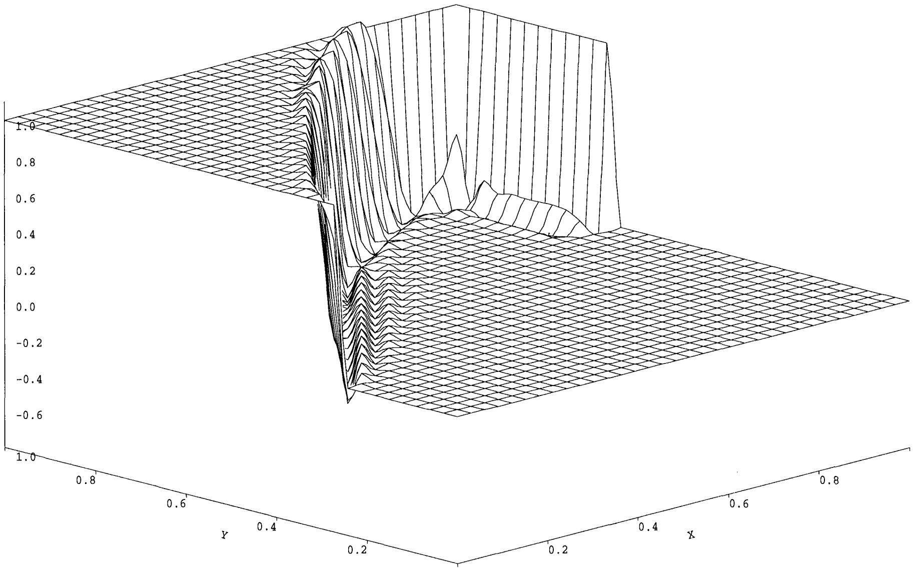


FIG. 2a. *Streamline diffusion FEM approximation to problem* (4.1).

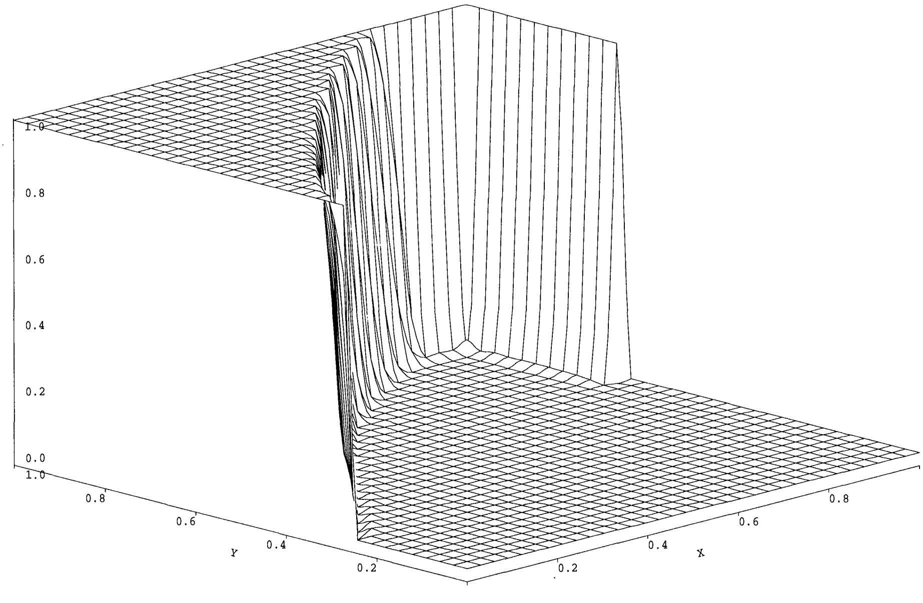


FIG. 2b. *Oscillation absorption FEM approximation to problem* (4.5).

A typical choice of $a(\cdot, \cdot)$ will be the symmetric part of $B^h(\cdot, \cdot)$,

$$a(u, v) = \frac{1}{2}(B^h(u, v) + B^h(v, u)),$$

or the $L^2(\Omega)$-inner product.

LEMMA 5.1. *Let $\mathbb{K}^h$ be closed and convex in X^h. Then $\Pi^h w^h$ exists and is unique.*

Proof. This follows immediately from the fact that a closed convex set in a Hilbert space has a unique element of minimal norm. □

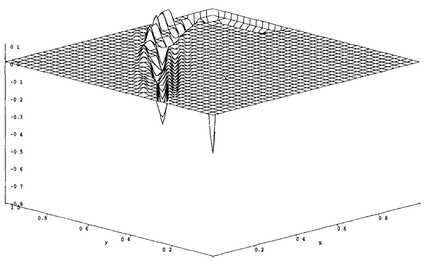


FIG. 2c. *Overshoot and undershoot in* 2a.

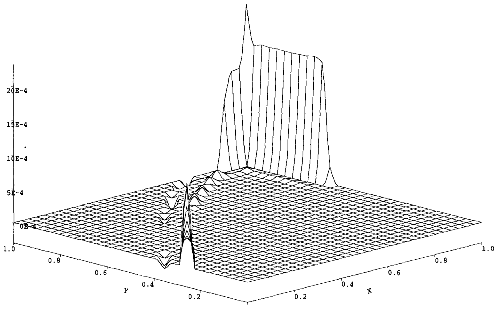


FIG. 2d. *Final nonlinear absorption term.*

$\Pi^h w^h$ is characterized by the following variational inequality: given $w^h \in X^h$, seek $\Pi^h w^h \in \mathbb{K}^h$ satisfying

$$a(w^h - \Pi^h w^h, \Pi^h w^h - v) \geq 0 \qquad \forall\, v \in \mathbb{K}^h. \tag{5.2}$$

Since $a(\cdot,\cdot)$ is symmetric, this variational inequality can be solved quite efficiently by a projected iterative method; see, e.g., Glowinski, Lions, Trémolières [7]. We propose computing the projection of u^h on $\mathbb{K}^h$ by means of a truncated projective iterative method. First note that $\Pi^h u^h$, defined by (5.2), has an error optimality property.

PROPOSITION 5.1. *Let $u^h \in X^h \subset X$ be an approximation to $u \in \mathbb{K} \subset X$. Suppose $\mathbb{K}$ is a closed, convex set and $\mathbb{K}^h := X^h \cap \mathbb{K}$ is nonempty. Then,*

$$\|u - \Pi^h u^h\|_A \leq 2\|u - u^h\|_A + \inf_{\chi \in \mathbb{K}^h} \|u - \chi\|_A.$$

Proof.

$$\begin{aligned} \|u - \Pi^h u^h\|_A &\leq \|u - u^h\|_A + \|u^h - \Pi^h u^h\|_A \\ &\leq \|u - u^h\|_A + \|u^h - \chi\|_A \end{aligned}$$

for any $\chi \in \mathbb{K}_h$

$$\leq \|u - u^h\|_A + \|u^h - u\|_A + \|u - \chi\|_A. \qquad \square$$

Remark 5.1. In the generality of the previous proposition, neither $\|u - u^h\|_A$ nor $\inf_{\chi \in \mathbb{K}^h} \|u - \chi\|_A$ may be omitted from the RHS of the inequality. This can be seen quite easily in planar examples where X^h is a line and K^h is an ellipse.

However, there are quite a few special situations in which the infimum over K^h may be replaced by an infimum over X^h. This is clear, e.g., for conforming linear finite element spaces; see Mosco and Strang [16]. For more general finite element spaces X^h, we refer to Lemma 3.2.

Concerning the calculation of $\Pi^h u^h$, note that since $a(\cdot,\cdot)$ is symmetric and positive definite, $\Pi^h u^h$ can be calculated using a projected iterative method. Letting $\{\phi_i\}$ be a basis for X^h and $A_{i,j} = a(\phi_i, \phi_j)$, suppose the constraint for $w^h = \sum_{j=1}^N c_j \phi_j(x)$ to be in $\mathbb{K}^h$ is equivalent to the coefficient bounds

$$\alpha_j \leq c_j \beta_j.$$

This holds for the standard nodal basis for conforming linears with $\alpha_j = u_{\min}$ and $\beta_j = u_{\max}$, since in that case $\phi_j(x)$ satisfies $0 \leq \phi_j(x) \leq 1$. With the pointwise projection $P_N : \mathbb{R}^N \mapsto$

($\prod_{j=1}^{N}[\alpha_j, \beta_j]$) defined by

$$(P_N x)_j = \begin{cases} x_j & \text{if } \alpha_j . \leq x_j \leq \beta_j, \\ \alpha_j & \text{if } x_j \leq \alpha_j, \\ \beta_j & \text{if } x_j \geq \beta_j, \end{cases}$$

the iterative method (projected successive overrelaxation) where $u^h = \sum d_j \phi_j$ and $\vec{f} = A\vec{d}$ now reads as follows: Given $\omega \in (0, 2)$, $\vec{c}^0 = (c_1^0, c_2^0, \ldots, c_N^0)^t$ satisfying $\alpha_j \leq c_j^0 \leq \beta_j$ define $\vec{c}^{n+1}$, $n = 0, 1, \ldots$, by

$$\text{(5.3)} \qquad \begin{cases} c_i^{n+\frac{1}{2}} = -\frac{1}{a_{i,i}} \left[\sum_{j=1}^{i-1} a_{i,j} c_j^{n+\frac{1}{2}} + \sum_{j=i+1}^{N} a_{i,j} c_j^n - f_i \right], \\ c_j^{n+1} = P_N \left[(1-\omega) c_i^n + \omega c_i^{n+\frac{1}{2}} \right]. \end{cases}$$

This is very cheap and produces an approximation in $\mathbb{K}^h$ which is close to u^h even when a few (O(1)) iterations are performed. Therefore we propose postprocessing u^h to get an approximate $\Pi^h u^h$ by applying O(1) iterations of (5.3) to (5.2). We will report on this later.

6. Conclusions. We have considered a special purpose numerical technique which is adapted to convective transport problems. Adding a nonlinear absorption term to the variational formulation of a convective transport problem effectively eliminates overshoots and undershoots in the approximate solution. With the streamline diffusion method as a basic discretization, the introduced nonlinearity causes very little increase in computational complexity. With the usual (centered) Galerkin discretization, the greatest computational challenge is simply finding a good preconditioner for the linearized problem. The usefulness of the variational inequality approach is currently limited by the lack of good iterative methods for highly nonsymmetric variational inequalities. One possible way around this difficulty is to postprocess the approximate solution using by symmetric variational inequalities. The latter are easily solvable using projective iterative methods.

REFERENCES

[1] O. Axelsson, *A generalized conjugate gradient, least square method*, Numer. Math., 51 (1987), pp. 209–227.

[2] O. Axelsson, V. Eijkhout, B. Polman, and P. Vassilevski, *Iterative solution of singular perturbation 2nd order boundary value problems by use of incomplete block-matrix factorization methods*, BIT, 29 (1989), pp. 867–889.

[3] J. Boland and W. Layton, *Error analysis of finite element methods for steady natural convection problems*, Numer. Funct. Anal. Optim., 11 (1990), pp. 449–483.

[4] H. Deconinck, K. G. Powell, P. L. Roe, and R. Struijs, *Multi dimensional schemes for scalar advection*, in AIAA Ed. Ser., 91 1532, AIAA, Washington, DC, May 1991.

[5] J. Dennis and R. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Ciffs, NJ, 1983.

[6] R. S. Falk, *Error estimates for the approximation of a class of variational inequalities*, Math. Comp., 28 (1974), pp. 963–971.

[7] R. Glowinski, J. L. Lions, and R. Trémolières, *Numerical analysis of variational inequalities*, in Studies in Mathematics and its Applications, Vol. 8, North-Holland, Amsterdam, 1981.

[8] T. Hughes and A. Brooks, *A multidimensional upwind scheme with no crosswind diffusion*, in AMD vol. 34, Finite element methods for convection dominated flows, T. J. Hughes, ed., ASME, New York, 1979.

[9] T. Hughes, M. Mallet, and A. Mizukami, *A new finite element formulation for computational fluid dynamics: II Beyond SUPG*, Comput. Methods Appl. Mech. Engrg., 54 (1986), pp. 341–355.

[10] C. Johnson, *Numerical Solution of Partial Differential Equations by the Finite Element Method*, Cambridge University Press, Cambridge, 1987.

[11] D. Kinderlehrer and G. Stampacchia, *An Introduction to Variational Inequalities and their Applications*, Academic Press, New York, San Diego, CA, 1980.

[12] W. LAYTON, J. MAUBACH, AND P. RABIER, *Uniform convergence estimates for an elementwise parallel finite element method*, I.C.M.A. Report, Department of Mathematics and Statistics, University of Pittsburgh, Pittsburgh, PA, 1992.

[13] W. LAYTON AND P. RABIER, *Domain decomposition via operator splitting for nonsymmetric problems*, Appl. Math. Lett., 5 (1992), pp. 67–70.

[14] G. MINTY, *Monotone (nonlinear) operators in Hilbert space*, Duke Math. J., 29 (1962), pp. 341–346.

[15] A. MIZUKAMI AND T. HUGHES, *A Petrov Galerkin finite element method for convection dominated flows: An accurate upwinding technique for satisfying the maximum principle*, Comput. Methods Appl. Mech. Engrg., 50 (1985), pp. 181–193.

[16] U. MOSCO AND G. STRANG, *One-sided approximation and variational inequalities*, Bull. Amer. Math. Soc., 80 (1974), pp. 308–312.

[17] R. STRUIS, H. DECONINCK, AND P. L. ROE, *Fluctuation splitting schemes for multidimensional convection problems: An alternative to finite volume and finite element methods*, Comput. Fluid Dynamics, 3 (1990).

[18] WING HUNG WONG, *On constrained multivariate splines and their approximation*, Numer. Math., 43 (1984), pp. 141–152.

SIAM J. SCI. COMPUT.
Vol. 17, No. 6, pp. 1347–1368, November 1996

QUASI-LAGUERRE ITERATION IN SOLVING SYMMETRIC TRIDIAGONAL EIGENVALUE PROBLEMS*

QIANG DU†, MING JIN‡, T. Y. LI†, AND Z. ZENG§

Abstract. In this article, the quasi-Laguerre iteration is established in the spirit of Laguerre's iteration for solving polynomial f with all real zeros. The new algorithm, which maintains the monotonicity and global convergence of the Laguerre iteration, no longer needs to evaluate f''. The ultimate convergence rate is $\sqrt{2}+1$. When applied to approximate the eigenvalues of a symmetric tridiagonal matrix, the algorithm substantially improves the speed of Laguerre's iteration.

Key words. eigenvalue, quasi-Laguerre iteration, symmetric tridiagonal matrix

AMS subject classification. 65F15

1. Introduction.

The globally and monotonically convergent Laguerre iteration

$$L_{\pm}(x) = x + \frac{n}{\left(-\frac{f'(x)}{f(x)}\right) \pm \sqrt{(n-1)\left[(n-1)\left(-\frac{f'(x)}{f(x)}\right)^2 - n\left(\frac{f''(x)}{f(x)}\right)\right]}} \tag{1}$$

has been successfully used to find the eigenvalues of an $n \times n$ symmetric tridiagonal matrix T with nonzero subdiagonal entries by approximating the zeros, always real and simple, of its characteristic polynomial

$$f(\lambda) = \det[T - \lambda I]. \tag{2}$$

Remarkable numerical results, in terms of both speed and accuracy, on a substantial variety of matrices have been obtained [10]. The algorithm employs the *split–merge* process, similar to Cuppen's divide-and-conquer strategy, which provides an excellent set of starting values that make the algorithm naturally parallel. Impressive speedups of the parallel version of the algorithm were reported in [15]. For $2,000 \times 2,000$ random matrices, the algorithm can reach a speedup of 52 on an N-cube with 64 nodes.

Laguerre's iteration, or modified Laguerre's iteration when clusters occur, converges ultimately with a very fast cubic convergence rate. Nonetheless, the most important advantage of Laguerre's iteration in solving (2), in contrast to Newton's iteration, is its monotone convergence. However, compared with Newton's iteration, a major disadvantage of Laguerre's iteration is the evaluation of f'', which is relatively time consuming. The purpose of this paper is to design a new algorithm in the spirit of Laguerre's iteration for finding zeros of (2). The algorithm, without the requirement of evaluating f'', maintains the monotonicity of Laguerre's iteration and converges globally with ultimate convergence rate $\sqrt{2}+1$ in finding zeros of any polynomial with all real zeros.

The formula (1) of Laguerre's iteration can be derived in diverse ways. The best one seems to be to answer the following question [9].

Question 1. Among all polynomials $p(x)$ of degree n with n real zeros and with $p(x_0) = f(x_0) \neq 0$, $p'(x_0) = f'(x_0)$, and $p''(x_0) = f''(x_0)$ at a specified real x_0, which one has a zero closest to x_0 and where?

*Received by the editors August 19, 1994; accepted for publication (in revised form) May 31, 1995.

†Department of Mathematics, Michigan State University, East Lansing, MI 48824-1027 (du@math.msu.edu, li@math.msu.edu). The first author's research was supported in part by DOE under grant DE-FG02-93ER25172. The third author's research was supported in part by NSF under grant CCR-9024840.

‡Department of Mathematics, Lambuth University, Jackson, TN 38301 (jinm66@use.usit.net).

§Department of Mathematics, Northeastern Illinois University, Chicago, IL 60625 (uzzeng@uxa.ecn.bgu.edu).

In general, $L_+(x_0)$ in (1) gives the closest zero from the right of all those polynomials and $L_-(x_0)$ gives the closest one from the left.

To avoid the evaluation of f'', we revise the above optimization problem as follows.

Question 2. Given two specified reals $a < b$, among all polynomials $p(x)$ of degree n with n real zeros, none of which lie in $[a, b]$, and with

$$\frac{p'(a)}{p(a)} = \frac{f'(a)}{f(a)} \quad \text{and} \quad \frac{p'(b)}{p(b)} = \frac{f'(b)}{f(b)}, \tag{3}$$

which one has a zero closest to a from the right or from the left and where?

The optimization problem for this class of polynomials can be easily solved and the solution has a closed form. To convert the solution of this optimization problem to an iterative scheme, which we call the *quasi-Laguerre iteration*, with the assumption that $x^{(k-1)}$ and $x^{(k)}$ are available, we may

(i) let $a = x^{(k-1)}$, $b = x^{(k)}$, and $x^{(k+1)}$ be the closest zero to $x^{(k)}$ from the right when $x^{(k-1)} < x^{(k)}$ and a zero of f is bigger than $x^{(k)}$,

(ii) let $a = x^{(k)}$, $b = x^{(k-1)}$, and $x^{(k+1)}$ be the closest zero to $x^{(k)}$ from the left when $x^{(k-1)} > x^{(k)}$ and a zero of f is less than $x^{(k)}$.

It is clear that the sequence so generated will converge to a zero of f monotonically.

The optimization problem in Question 2 is formulated in a more general form in §2 to account for multiple zeros. The ultimate rate of convergence of our algorithm so obtained is $\sqrt{2}+1$.

Our main goal here is to use the newly derived quasi-Laguerre iteration to approximate eigenvalues of symmetric tridiagonal matrices. To shorten our presentation, we only outline the derivation of our algorithm in this article and skip most of the theoretical details, including the tedious proof of its convergence properties. They are regrouped in [3] for interested readers.

The implementation details of our algorithm are described in §§4 and 5, and comprehensive numerical results on diverse types of matrices will be shown in §6. In general, our algorithm considerably improves the speed of Laguerre's iteration. Like Laguerre's iteration, our algorithm is inherently parallel and has a great capacity for vectorization. An intensive experiment in this regard will be reported in a future article.

During the writing of this paper, we became aware of an earlier unpublished work of Foster [5] in which a class of globally convergent iterations, including our quasi-Laguerre iteration for the case of simple roots, was studied. Our work here is based on a different approach and achieves much more general results.

2. Derivation of the quasi-Laguerre iteration. Let f be a polynomial of degree n with all of its zeros being real. For $a < b$ with $f(a)f(b) \neq 0$, let $\mathcal{F}$ be the class of polynomials $p(x)$ of degree n that satisfy the following conditions:

(i) all zeros of $p(x)$ are real,

(ii) none of the zeros of $p(x)$ are in $[a, b]$,

(iii) $p(a)p(b) \neq 0$ and

$$\frac{p'(a)}{p(a)} = \frac{f'(a)}{f(a)} \equiv q(a), \quad \frac{p'(b)}{p(b)} = \frac{f'(b)}{f(b)} \equiv q(b). \tag{4}$$

We pose here a more general optimization problem.

Question 3. For $p \in \mathcal{F}$, consider its mth $(m < n)$ zero to the right (or to the left) of a. Which polynomial in $\mathcal{F}$ has the closest one to a from the right (or from the left) and where?

To answer this question, let $z_1, z_2, \ldots, z_n$ be all the real zeros of a given polynomial $p(x)$ in $\mathcal{F}$ such that

$$z_1 \leq z_2 \leq \cdots \leq z_k < a < b < z_{k+1} \leq \cdots \leq z_n. \tag{5}$$

It follows from (4) that

$$\frac{p'(a)}{p(a)} = \sum_{i=1}^{n} \frac{1}{a - z_i} = q(a) \quad \text{and} \quad \frac{p'(b)}{p(b)} = \sum_{i=1}^{n} \frac{1}{b - z_i} = q(b). \tag{6}$$

For $1 \le i \le n$, let

$$X_i \equiv \frac{b - z_i}{a - z_i} = 1 + \frac{b - a}{a - z_i}.$$

Then, $X_i > 0$ for all i since no z_i falls between a and b. Moreover,

$$\begin{aligned} \sum_{i=1}^{n} X_i &= \sum_{i=1}^{n} \left(1 + \tfrac{b-a}{a-z_i}\right) = n + (b-a)q(a) \equiv r(a,b), \\ \sum_{i=1}^{n} \frac{1}{X_i} &= \sum_{i=1}^{n} \left(1 + \tfrac{a-b}{b-z_i}\right) = n + (a-b)q(b) = r(b,a), \end{aligned} \tag{7}$$

and from (5),

$$X_k \ge X_{k-1} \ge \cdots \ge X_1 > 1 > X_n \ge X_{n-1} \ge \cdots \ge X_{k+1} > 0. \tag{8}$$

For convenience, we rewrite (8) as

$$W_1 \ge W_2 \ge \cdots \ge W_k > 1 > W_{k+1} \ge \cdots \ge W_n \tag{9}$$

with

$$W_l = \begin{cases} X_{k+1-l} & \text{for } l \le k, \\ X_{n+k+1-l} & \text{for } l > k. \end{cases} \tag{10}$$

From (5), the mth zero of $p(x)$ to the left of a is z_{k-m+1} which, from (10), corresponds to W_m in (9), and minimizing $(a - z_{k-m+1})$ becomes maximizing W_m. Similarly, the mth zero of $p(x)$ to the right of a is z_{k+m} and minimizing $(z_{k+m} - a)$ becomes minimizing W_{n-m+1}. As a convention, if $m > k$ then the mth zero of p to the left of a will be taken as the $(n-m+1)$st zero to the right of a and if $m > n-k$ then the mth zero of p to the right of a is the $(n-m+1)$st zero to the left of a.

The optimization problem in Question 3 can now be converted to the following optimizations:

(P1) $$\text{Max } W_m$$

subject to

(11)
$$\sum_{i=1}^{n} W_i = r(a,b),$$
$$\sum_{i=1}^{n} \frac{1}{W_i} = r(b,a),$$
$$W_1 \ge W_2 \ge \cdots \ge W_n > 0,$$

(P2) $$\text{Min } W_{n-m+1}$$

subject to

$$\sum_{i=1}^{n} W_i = r(a,b),$$
$$\sum_{i=1}^{n} \frac{1}{W_i} = r(b,a),$$
$$W_1 \ge W_2 \ge \cdots \ge W_n > 0.$$

The solution of (P1) gives the mth closest zero to a from the left while the solution of (P2) gives the mth closest zero to a from the right.

For any $W = (W_1, \ldots, W_n)$ satisfying (11), it can be shown [3] that W_m satisfies

$$mr(b,a)W_m^2 - \left[r(a,b)r(b,a) - n^2 + 2mn\right] W_m + mr(a,b) \leq 0, \tag{12}$$

and the equality holds iff $W_j = W_m$ for all $j < m$ and $W_l = W_n$ for all $l > m$.

Similarly, W_{n-m+1} satisfies the same inequality, that is,

$$mr(b,a)W_{n-m+1}^2 - \left[r(a,b)r(b,a) - n^2 + 2mn\right] W_{n-m+1} + mr(a,b) \leq 0, \tag{13}$$

and the equality holds iff $W_j = W_{n-m+1}$ for all $j > n-m+1$ and $W_l = W_1$ for all $l < n-m+1$ [3].

Now, taking equalities in (12) and (13) yields the quadratic equation

$$mr(b,a)Y^2 - \left[r(a,b)r(b,a) - n^2 + 2mn\right] Y + mr(a,b) = 0.$$

Its solutions

$$Y_{m\pm} = \frac{r(a,b)r(b,a) - n^2 + 2mn \pm \sqrt{\left[r(a,b)r(b,a) - n^2\right]\left[r(a,b)r(b,a) - (n-2m)^2\right]}}{2mr(b,a)}$$

are both positive [3]. On the other hand, it follows from (12) and (13) that

$$Y_{m-} \leq W_m \leq Y_{m+} \tag{14}$$

and

$$Y_{m-} \leq W_{n-m+1} \leq Y_{m+}. \tag{15}$$

Accordingly, $\overline{W} = (\overline{W}_1, \ldots, \overline{W}_n)$ with $\overline{W}_1 = \cdots = \overline{W}_m = Y_{m+}$ and $\overline{W}_{m+1} = \cdots = \overline{W}_n$ is the solution of the maximization problem (P1) and, from (11),

$$mY_{m+} + (n-m)\overline{W}_n = r(a,b),$$

$$\frac{m}{Y_{m+}} + \frac{n-m}{\overline{W}_n} = r(b,a).$$

Solving for $\overline{W}_n$, one can easily see that $\overline{W}_n$ is actually $Y_{(n-m)-}$. So,

$$\overline{W}_j = \begin{cases} Y_{m+}, & j \leq m, \\ Y_{(n-m)-}, & j > m. \end{cases} \tag{16}$$

Similarly, the solution of the minimization problem (P2) is $\underline{W} = (\underline{W}_1, \ldots, \underline{W}_n)$ with

$$\underline{W}_j = \begin{cases} Y_{(n-m)+}, & j < n-m+1, \\ Y_{m-}, & j \geq n-m+1. \end{cases} \tag{17}$$

Recall that W_j's are defined by

$$W_j = 1 + \frac{b-a}{a-u_j}, \qquad j = 1, \ldots, n,$$

where u_j's are zeros of a certain polynomial in $\mathcal{F}$. Thus any polynomial whose zeros satisfy (16) must be in the form

$$p_1(x) = C(x - u_{m-})^m (x - v_1)^{n-m}, \tag{18}$$

where C is a constant and

$$1 + \frac{b-a}{a - u_{m-}} = Y_{m+} \quad \text{or, equivalently,} \quad u_{m-} = a - \frac{b-a}{Y_{m+} - 1} \tag{19}$$

and

$$1 + \frac{b-a}{a - v_1} = Y_{(n-m)-} \quad \text{or, equivalently,} \quad v_1 = a - \frac{b-a}{Y_{(n-m)-} - 1}.$$

Further, any polynomial whose zeros satisfy (17) must be in the form

$$p_2(x) = C(x - u_{m+})^m (x - v_2)^{n-m}, \tag{20}$$

where C is a constant and

$$1 + \frac{b-a}{a - u_{m+}} = Y_{m-} \quad \text{or, equivalently,} \quad u_{m+} = a - \frac{b-a}{Y_{m-} - 1} \tag{21}$$

and

$$1 + \frac{b-a}{a - v_2} = Y_{(n-m)+} \quad \text{or, equivalently,} \quad v_2 = a - \frac{b-a}{Y_{(n-m)+} - 1}.$$

From (19) and (21),

$$\frac{b - u_{m-}}{a - u_{m-}} = Y_{m+} > 0 \quad \text{and} \quad \frac{b - u_{m+}}{a - u_{m+}} = Y_{m-} > 0,$$

so both u_{m+} and u_{m-} are outside the interval (a, b). Similarly, both v_1 and v_2 are outside the interval (a, b) since $Y_{(n-m)\pm} > 0$. Consequently, both $p_1(x)$ and $p_2(x)$ are in $\mathcal{F}$.

Moreover, from (19) and (21)

$$\begin{aligned}
u_{m\pm} &= a - \frac{b-a}{Y_{m\mp} - 1} \\
&= a - \frac{2mr(b,a)(b-a)}{r(a,b)r(b,a) - n^2 + 2mn - 2mr(b,a) \mp \sqrt{[r(a,b)r(b,a) - n^2][r(a,b)r(b,a) - (n-2m)^2]}} \\
&= a + \frac{2m[n - (b-a)q(b)]}{-(b-a)R - 2mq(b) \pm \sqrt{R[(b-a)^2 R + 4m(n-m)]}},
\end{aligned} \tag{22}$$

where

$$q(a) = \frac{f'(a)}{f(a)}, \quad q(b) = \frac{f'(b)}{f(b)}, \quad \text{and} \quad R = n\left(\frac{q(a) - q(b)}{b-a}\right) - q(a)q(b).$$

The solutions of the optimization problems in Question 3 can now be described as follows:

(i) If f has at least m (counting multiplicities) zeros to the left of a, then the zero u_{m-} in (22) of the polynomial

$$p_1(x) = C(x - u_{m-})^m (x - v_1)^{n-m}$$

in (18) is the closest mth zero to the left of a among all polynomials in $\mathcal{F}$. It is also clear that

$$z_{m-} \leq u_{m-} \leq \cdots \leq u_{2-} \leq u_{1-} < a < b, \tag{23}$$

where z_{m-} is the mth zero of f to the left of a.

(ii) If f has at least m (counting multiplicities) zeros to the right of b, then the zero u_{m+} in (22) of the polynomial

$$p_2(x) = C(x - u_{m+})^m (x - v_2)^{n-m}$$

in (20) is the closest mth zero to the right of b among all polynomials in $\mathcal{F}$. It is also clear that

$$a < b < u_{1+} \leq u_{2+} \leq \cdots \leq u_{m+} \leq z_{m+}, \tag{24}$$

where z_{m+} is the mth zero of f to the right of b.

3. The quasi-Laguerre iteration and its rate of convergence. Let f be a polynomial of degree n with all its zeros being real. Let z_M be a zero of f with multiplicity $M \geq 1$. To approximate z_M from its left, we shall use u_{m+} in (22) with $m \leq M$ to generate a monotonically increasing sequence $\{x_{m+}^{(k)}\}$ which converges to z_M as $k \to \infty$. Similarly, to approximate z_M from its right, a monotonically decreasing sequence $\{x_{m-}^{(k)}\}$ converging to z_M, as $k \to \infty$, can be generated by using u_{m-} in (22).

To be more precise, suppose z_M is on the right-hand side of two starting points $x_{m+}^{(0)} < x_{m+}^{(1)}$ and none of the zeros of f lie between $x_{m+}^{(0)}$ and z_M. Then, for $k \geq 1$, we may let

$$\begin{aligned} x_{m+}^{(k+1)} &= x_{m+}^{(k-1)} + \\ &\frac{2m\left[n - (x_{m+}^{(k)} - x_{m+}^{(k-1)})q(x_{m+}^{(k)})\right]}{-(x_{m+}^{(k)} - x_{m+}^{(k-1)})R - 2mq(x_{m+}^{(k)}) + \sqrt{R\left[(x_{m+}^{(k)} - x_{m+}^{(k-1)})^2 R + 4m(n-m)\right]}}, \end{aligned} \tag{25}$$

where

$$q(x_{m+}^{(k-1)}) = \frac{f'(x_{m+}^{(k-1)})}{f(x_{m+}^{(k-1)})}, \quad q(x_{m+}^{(k)}) = \frac{f'(x_{m+}^{(k)})}{f(x_{m+}^{(k)})},$$

and

$$R = n\left(\frac{q(x_{m+}^{(k-1)}) - q(x_{m+}^{(k)})}{x_{m+}^{(k)} - x_{m+}^{(k-1)}}\right) - q(x_{m+}^{(k-1)})q(x_{m+}^{(k)}).$$

In other words, we replace a, b, and u_{m+} in (22) by $x_{m+}^{(k-1)}$, $x_{m+}^{(k)}$, and $x_{m+}^{(k+1)}$, respectively.

Remark. Interchanging $x_{m+}^{(k-1)}$ and $x_{m+}^{(k)}$, (25) can also be written as

$$\begin{aligned} x_{m+}^{(k+1)} &= x_{m+}^{(k)} + \\ &\frac{2m\left[n - (x_{m+}^{(k-1)} - x_{m+}^{(k)})q(x_{m+}^{(k-1)})\right]}{-(x_{m+}^{(k-1)} - x_{m+}^{(k)})R - 2mq(x_{m+}^{(k-1)}) + \sqrt{R\left[(x_{m+}^{(k-1)} - x_{m+}^{(k)})^2 R + 4m(n-m)\right]}}. \end{aligned} \tag{26}$$

From what has been derived in the last section, it is easy to see that

$$x_{m+}^{(k)} < x_{m+}^{(k+1)} \leq z_M$$

and no zeros of f fall between $x_{m+}^{(k+1)}$ and z_M. So the sequence $\{x_{m+}^{(k)}\}$ satisfies

$$x_{m+}^{(0)} < x_{m+}^{(1)} < \cdots < x_{m+}^{(k)} < \cdots \leq z_M.$$

Similarly, when two starting points $x_{m-}^{(1)} < x_{m-}^{(0)}$ are available on the right-hand side of z_M with no zeros of f lying in $(z_M, x_{m-}^{(0)})$, then for $k \geq 1$, let $a = x_{m-}^{(k)}$, $b = x_{m-}^{(k-1)}$, and $u_{m-} = x_{m-}^{(k+1)}$ in (22). Namely, let

$$x_{m-}^{(k+1)} = x_{m-}^{(k)} + \tag{27}$$
$$\frac{2m\left[n - (x_{m-}^{(k-1)} - x_{m-}^{(k)})q(x_{m-}^{(k-1)})\right]}{-(x_{m-}^{(k-1)} - x_{m-}^{(k)})R - 2mq(x_{m-}^{(k-1)}) - \sqrt{R\left[(x_{m-}^{(k-1)} - x_{m-}^{(k)})^2R + 4m(n-m)\right]}},$$

where

$$q(x_{m-}^{(k)}) = \frac{f'(x_{m-}^{(k)})}{f(x_{m-}^{(k)})}, \quad q(x_{m-}^{(k-1)}) = \frac{f'(x_{m-}^{(k-1)})}{f(x_{m-}^{(k-1)})},$$

and

$$R = n\left(\frac{q(x_{m-}^{(k)}) - q(x_{m-}^{(k-1)})}{x_{m-}^{(k-1)} - x_{m-}^{(k)}}\right) - q(x_{m-}^{(k)})q(x_{m-}^{(k-1)}).$$

Again it is clear that

$$z_M \leq x_{m-}^{(k+1)} < x_{m-}^{(k)}$$

and none of the zeros of f lie between z_M and $x_{m-}^{(k+1)}$. Moreover, the sequence $\{x_{m-}^{(k)}\}$ satisfies

$$z_M \leq \cdots < x_{m-}^{(k)} < \cdots < x_{m-}^{(1)} < x_{m-}^{(0)}.$$

We shall call the sequences $\{x_{m\pm}^{(k)}\}$ defined by (25) (or (26)) and (27) the *quasi-Laguerre iterations*. The convergence property of both sequences $\{x_{m\pm}^{(k)}\}$ is given in the following theorem.

THEOREM 3.1. *Both sequences $\{x_{m\pm}^{(k)}\}$ converge to z_M monotonically. Furthermore,*

(i) when $m < M$, the convergence is linear and the convergence ratio

$$\omega \equiv \lim_{k\to\infty} \frac{x_{m\pm}^{(k+1)} - z_M}{x_{m\pm}^{(k)} - z_M}$$

is the only real solution of

$$\frac{n-M-m}{n-M}x^3 - x^2 - x + \frac{M-m}{M} = 0$$

in $(0, 1)$.

(ii) when $m = M$, the convergence is superlinear with convergence rate $\sqrt{2}+1$.

Proof. See [3] for a complete proof of Theorem 3.1. □

4. Solving symmetric tridiagonal eigenvalue problems. In this section, we shall use the quasi-Laguerre iteration derived in previous sections to approximate all eigenvalues of symmetric tridiagonal matrices.

4.1. Evaluation of the logarithmic derivative f'/f of the determinant. Let T be a symmetric tridiagonal matrix of the form

$$T = [\beta_{i-1}, \alpha_i, \beta_i] = \begin{pmatrix} \alpha_1 & \beta_1 & & & & \\ \beta_1 & \alpha_2 & \beta_2 & & 0 & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & 0 & & \beta_{n-2} & \alpha_{n-1} & \beta_{n-1} \\ & & & & \beta_{n-1} & \alpha_n \end{pmatrix}. \tag{28}$$

We may assume, without loss of generality, that T is unreduced; that is, $\beta_j \neq 0$, $j = 1, \ldots, n-1$. For an unreduced T, the characteristic polynomial

$$f(\lambda) \equiv \det(T - \lambda I) \tag{29}$$

has only *real* and *simple* zeros [16, p. 300]. To use our quasi-Laguerre iteration developed in previous sections for finding zeros of $f(\lambda)$, or the eigenvalues of T, it is necessary to evaluate f and f' efficiently with satisfactory accuracy in the first place.

It is well known that the characteristic polynomial $f(\lambda)$ and its derivative with respect to λ can be evaluated by three-term recurrences [16, p. 423]:

$$\begin{cases} \rho_0 = 1, \quad \rho_1 = \alpha_1 - \lambda, \\ \rho_i = (\alpha_i - \lambda)\rho_{i-1} - \beta_{i-1}^2 \rho_{i-2}, \qquad i = 2, 3, \ldots, n, \end{cases} \tag{30}$$

$$\begin{cases} \rho_0' = 0, \quad \rho_1' = -1, \\ \rho_i' = (\alpha_i - \lambda)\rho_{i-1}' - \rho_{i-1} - \beta_{i-1}^2 \rho_{i-2}', \qquad i = 2, 3, \ldots, n, \end{cases} \tag{31}$$

and

$$f(\lambda) = \rho_n, \quad f'(\lambda) = \rho_n'.$$

However, these recurrences may suffer from a severe underflow–overflow problem and require constant testing and scaling. The following modified recurrence equation [10] is the result of careful investigation of the problem and is more stable than the code presented in [12]. It computes the logarithmic derivative $q(\lambda) = f'(\lambda)/f(\lambda)$, required in the formulae (25)–(27). Let

$$\xi_i = \frac{\rho_i}{\rho_{i-1}}, \qquad \eta_i = -\frac{\rho_i'}{\rho_i},$$

$$\begin{cases} \xi_1 = \alpha_1 - \lambda, \\ \xi_i = \alpha_i - \lambda - \dfrac{\beta_{i-1}^2}{\xi_{i-1}}, \quad i = 2, 3, \ldots, n, \end{cases} \tag{32}$$

$$\begin{cases} \eta_0 = 0, \quad \eta_1 = \dfrac{1}{\xi_1}, \\ \eta_i = \dfrac{1}{\xi_i}\left[(\alpha_i - \lambda)\eta_{i-1} + 1 - \left(\dfrac{\beta_{i-1}^2}{\xi_{i-1}}\right)\eta_{i-2}\right], \quad i = 2, 3, \ldots, n, \end{cases} \tag{33}$$

and

$$-\frac{f'(\lambda)}{f(\lambda)} = \eta_n.$$

To prevent the algorithm from breaking down when $\xi_i = 0$ for some $1 \le i \le n$, an extra check is provided:

- If $\xi_1 = 0$ (i.e., $\alpha_1 = \lambda$), set $\xi_1 = \beta_1^2 \varepsilon^2$;
- If $\xi_i = 0, i > 1$, set $\xi_i = \frac{\beta_{i-1}^2 \varepsilon^2}{\xi_{i-1}}$;

where ε is the machine precision. A determinant evaluation subroutine DETEVL has been implemented [10] according to the recurrences (32) and (33). When ξ_i, $i = 1, \ldots, n$ are known, the Sturm sequence is available [13, p. 47]. Thus, as a by-product, DETEVL also evaluates the number of eigenvalues of T which are less than λ.

Let $\lambda_1 < \lambda_2 < \cdots < \lambda_n$ be the zeros of $f(\lambda)$ and $\hat{\lambda}_1 < \hat{\lambda}_2 < \cdots < \hat{\lambda}_n$ be the zeros of the numerical approximation $\hat{f}(\lambda)$. It was shown in [10] that

$$|fl(\hat{\lambda}_i) - \lambda_i| \le \frac{5\varepsilon}{2} \max_j\{|\beta_j| + |\beta_{j+1}|\} + |\lambda_i|\varepsilon. \tag{34}$$

4.2. The split–merge process. Let

$$\lambda_1 < \lambda_2 < \cdots < \lambda_n$$

be the zeros of f in (29). To use the quasi-Laguerre iteration to approximate any λ_i, $i = 1, 2, \ldots, n$, it is essential to provide a pair of starting points $x^{(0)}$ and $x^{(1)}$, being either $x^{(0)} < x^{(1)} < \lambda_i$ or $\lambda_i < x^{(1)} < x^{(0)}$, with no other λ_j's lying between $x^{(0)}$, $x^{(1)}$, and λ_i. For this purpose, we *split* the matrix T into

$$\hat{T} = \begin{pmatrix} T_0 & 0 \\ 0 & T_1 \end{pmatrix}, \tag{35}$$

where

$$T_0 = \begin{pmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{k-1} \\ & & \beta_{k-1} & \alpha_k - \beta_k \end{pmatrix}, \quad T_1 = \begin{pmatrix} \alpha_{k+1} - \beta_k & \beta_{k+1} & & \\ \beta_{k+1} & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{n-1} \\ & & \beta_{n-1} & \alpha_n \end{pmatrix}. \tag{36}$$

Obviously, the eigenvalues of $\hat{T}$ consist of eigenvalues of T_0 and T_1. Without loss of generality, we may assume $\beta_i > 0$ for all $i = 1, 2, \ldots, n-1$, since in (30)–(33), β_i's always appear in their square form. The following interlacing property for this rank-one tearing is important to our algorithm.

THEOREM 4.1. *Let $\lambda_1 < \lambda_2 < \cdots < \lambda_n$ and $\hat{\lambda}_1 \le \hat{\lambda}_2 \le \cdots \le \hat{\lambda}_n$ be eigenvalues of T and $\hat{T}$, respectively. Then*

$$\hat{\lambda}_1 \le \lambda_1 \le \hat{\lambda}_2 \le \lambda_2 \le \cdots \le \hat{\lambda}_n \le \lambda_n < \hat{\lambda}_{n+1}$$

with the convention $\hat{\lambda}_{n+1} = \hat{\lambda}_n + 2\beta_k$.

Proof. See [6, Thm. 8.6.2, p. 462] for a complete proof of Theorem 4.1. □

The eigenvalues of $\hat{T}$ will be used critically to approximate the eigenvalues of T by our quasi-Laguerre iteration. We shall call this procedure, splitting T into T_0 and T_1 of $\hat{T}$ and

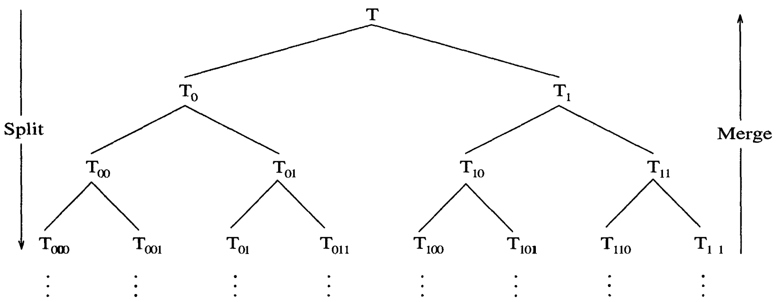


FIG. 1. *Split–merge processes.*

using eigenvalues of $\hat{T}$, consisting of eigenvalues of T_0 and T_1, to approximate eigenvalues of T, the *split–merge* process, similar to Cuppen's divide-and-conquer strategy [2, 4, 8].

To evaluate certain eigenvalues λ_i, $i = 1, 2, \ldots, n$, of T, both $\hat{\lambda}_i$ and $\hat{\lambda}_{i+1}$ are suitable candidates for the first starting point of our algorithm, since by Theorem 4.1,

$$\hat{\lambda}_i \leq \lambda_i \leq \hat{\lambda}_{i+1}.$$

For the second starting point, let

$$c = \frac{\hat{\lambda}_i + \hat{\lambda}_{i+1}}{2}$$

and evaluate $\frac{f'(c)}{f(c)}$ by the subroutine DETEVL [10]. As mentioned before, the Sturm sequence at c, which decides the position of c relative to λ_i, is a by-product of this evaluation. When $c > \lambda_i$, we let $x_-^{(0)} = \hat{\lambda}_{i+1}$, $x_-^{(1)} = c$ and when $c < \lambda_i$, we let $x_+^{(0)} = \hat{\lambda}_i$ and $x_+^{(1)} = c$. Then $\lambda_i < x_-^{(1)} < x_-^{(0)}$ or $x_+^{(0)} < x_+^{(1)} < \lambda_i$. Respectively, equations (26) and (27) with $m = 1$ are used to generate $x_\pm^{(k+1)}$ for $k \geq 1$, namely,

$$x_\pm^{(k+1)} = x_\pm^{(k)} + \frac{2\left[n - (x_\pm^{(k-1)} - x_\pm^{(k)})q(x_\pm^{(k-1)})\right]}{-(x_\pm^{(k-1)} - x_\pm^{(k)})R - 2q(x_\pm^{(k-1)}) \pm \sqrt{R\left[(x_\pm^{(k-1)} - x_\pm^{(k)})^2 R + 4(n-1)\right]}}.$$

Moreover sequences $\{x_\pm^{(k)}\}_{k=1}^{\infty}$ are obtained which converge monotonically to λ_i with ultimate convergence rate $\sqrt{2} + 1$ by Theorem 3.1.

The eigenvalues of $\hat{T}$ in (35) consist of eigenvalues of T_0 and T_1 in (36). To find eigenvalues of T_0 and T_1, the split–merge process described above may be applied again. Indeed, the splitting process can be applied to T recursively (See Figure 1) until 2×2 and 1×1 matrices are reached.

After T is well split into a tree structure as shown in Figure 1, the merging process in the reverse direction from 2×2 and 1×1 matrices can be started. More specifically, let T_σ be split into $T_{\sigma 0}$ and $T_{\sigma 1}$. Let $\hat{\lambda}_1^\sigma, \ldots, \hat{\lambda}_m^\sigma$ be eigenvalues of $\hat{T}_\sigma = \begin{pmatrix} T_{\sigma 0} & 0 \\ 0 & T_{\sigma 1} \end{pmatrix}$ in ascending order. Then the quasi-Laguerre iteration is applied to the polynomial equation

$$f_\sigma(\lambda) \equiv \det[T_\sigma - \lambda I] = 0$$

from either $\hat{\lambda}_i^{(\sigma)}$ or $\hat{\lambda}_{i+1}^{(\sigma)}$ to obtain the corresponding eigenvalue λ_i^σ, $i = 1, 2, \ldots, m$, by the merging process described above. This process continues until T_0 and T_1 are merged into T. That is, in the final step all the eigenvalues of T are obtained by applying the quasi-Laguerre iteration to $f(\lambda) = \det(T - \lambda I)$ from eigenvalues of T_0 and T_1.

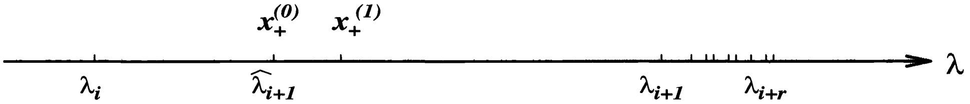


FIG. 2. *Converging to relatively closer eigenvalues.*

4.3. Cluster and deflation. By Theorem 4.1, $\lambda_i \in (\hat{\lambda}_i, \hat{\lambda}_{i+1})$ for each $i = 1, \dots, n$ with the convention $\hat{\lambda}_{n+1} = \hat{\lambda}_n + 2\beta_k$. If $\hat{\lambda}_{i+1} - \hat{\lambda}_i$ is less than the error tolerance, then either $\hat{\lambda}_i$ or $\hat{\lambda}_{i+1}$ can be accepted as λ_i. In general, if $\hat{T}$ has a cluster of $r + 1$ very close eigenvalues, for instance, $\hat{\lambda}_{j+r} - \hat{\lambda}_j$ is less than the error tolerance for certain $1 \le j \le n - r$, then r eigenvalues $\lambda_j, \lambda_{j+1}, \dots, \lambda_{j+r-1}$ of T can be obtained free of computations. They can be set to any one of $\hat{\lambda}_j, \dots, \hat{\lambda}_{j+r}$.

Since the matrix T in (28) is unreduced, its eigenvalues are all simple. Therefore using $m = 1$ in the quasi-Laguerre iterations given in (26) or (27) seems appropriate in all cases to obtain ultimate superlinear convergence with convergence rate $\sqrt{2} + 1$. However, in some cases, there may exist a group of $r > 1$ eigenvalues of T, say,

$$\lambda_{i+1} < \lambda_{i+2} < \cdots < \lambda_{i+r}$$

which are relatively close to each other, compared to the distance from the two starting points, say $x_+^{(0)} < x_+^{(1)} < \lambda_{i+1}$, as shown in Figure 2. Some of them may even be numerically indistinguishable.

Numerical evidence shows that to reach λ_{i+1} from $x_+^{(0)} (= \hat{\lambda}_{i+1})$ and $x_+^{(1)}$, it may take many steps of the quasi-Laguerre iteration with $m = 1$ before showing super-linear convergence. In this situation, the quasi-Laguerre iteration with $m = r$ in (26) may be used to speed up the convergence. Exhibited in (24), the sequences $\{x_{m+}^{(k)}\}$ in (26) with different m's satisfy

$$x_+^{(k-1)} < x_{1+}^{(k)} < x_{2+}^{(k)} < \cdots < x_{r+}^{(k)} < \lambda_{i+r}.$$

Accordingly, $x_{r+}^{(k)}$ is relatively closer to λ_{i+1}. Therefore if the number r of those relatively closer zeros can be estimated, then instead of using $m = 1$ as usual, we may let $x_+^{(k)} = x_{r+}^{(k)}$. To estimate r in our algorithm, we put λ_j in the same group with λ_{i+1} if

$$|\hat{\lambda}_j - \hat{\lambda}_{i+2}| < 0.01|\hat{\lambda}_{i+2} - \hat{\lambda}_{i+1}|.$$

It might happen that $\lambda_{i+1} < x_+^{(k)}$, which can be revealed by the information of the Sturm sequence given at $x_+^{(k)}$; in this case, the number r may be reduced. Therefore an overestimate of the number r causes essentially no harm because our algorithm can dynamically reduce r, while an underestimate may result in slow convergence.

4.4. Partial spectrum. From the strong interlacing property given in Theorem 4.1 one can easily obtain the following.

PROPOSITION 4.2. *If $[a, b]$ contains k eigenvalues of $\hat{T}$, then $[a, b]$ contains at least $k - 1$ and at most $k + 1$ eigenvalues of T. More precisely, let $\kappa(x)$ be the number of eigenvalues of T which are less than $x \in \mathbb{R}$ and $\hat{\lambda}_{s+1}, \dots, \hat{\lambda}_{s+k}$ be all the eigenvalues of $\hat{T}$ in $[a, b]$. Then $s - 1 \le \kappa(a) \le s$ and $s + k - 1 \le \kappa(b) \le s + k$.*

In our algorithm, either $\hat{\lambda}_i$ or $\hat{\lambda}_{i+1}$ of $\hat{T}$ is used as the first starting point of our quasi-Laguerre iteration to approximate λ_i of T. To find eigenvalues of T in a given interval $[a, b]$, the eigenvalues of $\hat{T}$ in $[a, b]$ are known, say $\hat{\lambda}_{s+1}, \dots, \hat{\lambda}_{s+k}$. By evaluating $\kappa(a)$ and $\kappa(b)$, the actual number of eigenvalues of T in $[a, b]$ is $\sigma = \kappa(b) - \kappa(a)$. Hence

$\lambda_{\kappa(a)+1}, \ldots, \lambda_{\kappa(a)+\sigma}$ are the eigenvalues of T in $[a, b]$. By Proposition 4.2, $s-1 \le \kappa(a) \le s$ and $s+k-1 \le \kappa(b) \le s+k$, so σ can either be $k-1$, k, or $k+1$. Thus, at most $k+2$ values are needed to be considered as the first starting points to evaluate these σ eigenvalues of T. Let

$$\hat{\hat{\lambda}}_s = a, \quad \hat{\hat{\lambda}}_{s+1} = \hat{\lambda}_{s+1}, \quad \hat{\hat{\lambda}}_{s+2} = \hat{\lambda}_{s+2}, \quad \ldots, \quad \hat{\hat{\lambda}}_{s+k} = \hat{\lambda}_{s+k}, \quad \hat{\hat{\lambda}}_{s+k+1} = b.$$

Then σ values among them can serve as the first starting points which will lead to all σ eigenvalues of T in $[a, b]$.

In some instances, the eigenvalues of T of interest are identified in magnitude, say the largest 20%, then they can be evaluated by using the largest 20% eigenvalues of $\hat{T}$ as starting points without computing the other 80% eigenvalues of $\hat{T}$ and T.

4.5. Stopping criteria. The following stopping criterion was suggested by Kahan [9]:

$$|x^{(k+1)} - x^{(k)}|^2 \le (|x^{(k)} - x^{(k-1)}| - |x^{(k+1)} - x^{(k)}|) = \tau, \tag{37}$$

where τ is the error tolerance. This criterion is based on the following observation. Let

$$q_k = \left| \frac{x^{(k+1)} - x^{(k)}}{x^{(k)} - x^{(k-1)}} \right|.$$

Then as $\{x^{(k)}\}_{k=1}^{\infty}$ converges to λ when $k \to \infty$, q_k is normally decreasing. Thus

$$\begin{aligned}|\lambda - x^{(k+1)}| &= \left| \sum_{i=0}^{\infty} (x^{(k+2+i)} - x^{(k+1+i)}) \right| \le \sum_{i=0}^{\infty} |x^{(k+2+i)} - x^{(k+1+i)}| \\ &\le |x^{(k+1)} - x^{(k)}| \sum_{i=1}^{\infty} q_k^i = \frac{q_k |x^{(k+1)} - x^{(k)}|}{1 - q_k} \\ &= \frac{|x^{(k+1)} - x^{(k)}|^2}{|x^{(k)} - x^{(k-1)}| - |x^{(k+1)} - x^{(k)}|}.\end{aligned}$$

From (34), an obvious error tolerance at λ_i can be chosen as

$$\tau = \frac{5\varepsilon}{2} \max_j \{|\beta_j| + |\beta_{j+1}|\} + |x^{(k+1)}|\varepsilon.$$

5. Practical considerations in implementation.

5.1. Initial points for the quasi-Laguerre iteration. At every stage of the merging process, we evaluate the eigenvalues $\lambda_1 < \cdots < \lambda_m$ of an $m \times m$ submatrix, given m initial values $\hat{\lambda}_1 \le \cdots \le \hat{\lambda}_m$ and an upper bound $\hat{\lambda}_{m+1}$ that interlace those m eigenvalues:

$$\hat{\lambda}_1 \le \lambda_1 \le \hat{\lambda}_2 \le \lambda_2 \le \ldots \le \hat{\lambda}_m \le \lambda_m \le \hat{\lambda}_{m+1}$$

(see Theorem 4.1). To evaluate an eigenvalue λ_i by the quasi-Laguerre iteration, two initial points, say $x^{(0)}$ and $x^{(1)}$, are required on the same side of λ_i without any other eigenvalues between them and $x^{(1)}$ is chosen to be closer to λ_i than $x^{(0)}$. As we described in §4.2, either $\hat{\lambda}_i$ or $\hat{\lambda}_{i+1}$ can be used as the first initial point $x^{(0)}$ because of the interlacing property:

$$\hat{\lambda}_i \le \lambda_i \le \hat{\lambda}_{i+1} \quad \text{for } i = 1, \ldots, m,$$

and bisection point $c = \frac{\hat{\lambda}_i + \hat{\lambda}_{i+1}}{2}$ is chosen as the second initial point $x^{(1)}$.

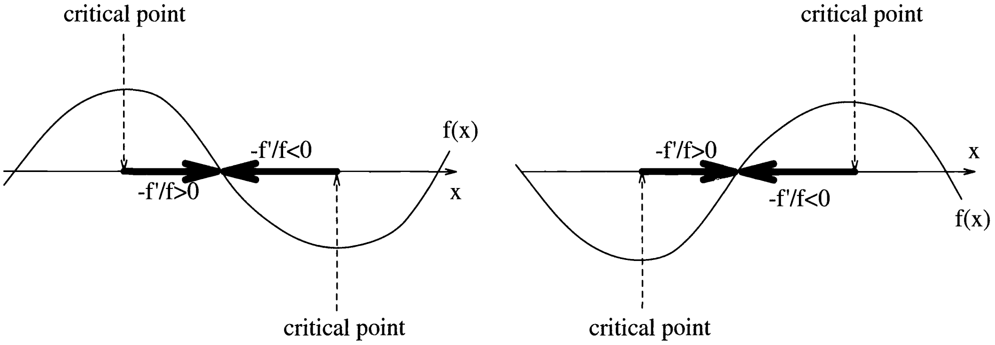


FIG. 3. *The two initial points should be between one of the nearby critical points of f and the zero. This can be identified by the sign pattern of $-f'/f$.*

While this approach of obtaining initial points is the simplest one, it may not be the most efficient. Several improvements are made in our practical implementation. First, since f'/f is evaluated at every initial value $\hat{\lambda}_j$ $(j = 1, \dots, m)$, one step of Newton's iteration can be performed without extra cost and the result can be considered as a candidate, besides c, for $x^{(1)}$. Second and more importantly, from our computing experience, the high-order convergence of the quasi-Laguerre iteration occurs only when there is no critical point of f (i.e., zero of f') between $x^{(0)}$ and λ_i (see Figure 3). In other words, if $x^{(0)}$ is to the left (resp., right) of λ_i, then it is desirable for $-f'(x^{(0)})/f(x^{(0)})$ to be positive (resp., negative). If there is at least one critical point in $[\hat{\lambda}_i, \hat{\lambda}_{i+1}]$, then bisection or one-step Newton iteration is used repeatedly until the above requirement at $x^{(0)}$ is satisfied. Based on these observations, our procedure for determining a pair of initial points among different choices to achieve the best possible efficiency is summarized in the algorithm INIPTS in Figure 4.

5.2. An alternative approach for clusters of eigenvalues. The multiplicity estimation and the quasi-Laguerre iterations (26) and (27) with $m > 1$ can successfully overcome the slow convergence when a cluster of eigenvalues occurs. An alternative approach without multiplicity estimation as well as dynamical adjustments of the multiplicities can also work very efficiently in practice.

By Theorem 3.1, when the quasi-Laguerre iteration for simple zero $(m = 1)$ is used to approximate an M-fold zero x^*, or a cluster of M simple zeros, of a polynomial $f(\lambda)$ of degree n, it converges linearly with a ratio $q_{n,M}$ which is the only solution of

$$\frac{n-M-1}{n-M}x^3 - x^2 - x + \frac{M-1}{M} = 0$$

in $(0, 1)$. Let $x^{(1)}, x^{(2)}, \dots, x^{(k)}, \dots$ be the monotone sequence of the quasi-Laguerre iterates. It can be shown [3] that $q_* \equiv q_{3,2}$, the zero of $x^2 + x - 0.5 = 0$ in $(0, 1)$, is the smallest linear ratio for multiple zeros. That is, $q_{n,M} \geq q_{3,2}$ when $n \geq 3$ and $M \geq 2$. When $\{x^{(k)}\}$ converges superlinearly, then

$$q_k \equiv \frac{|x^{(k+1)} - x^{(k)}|}{|x^{(k)} - x^{(k-1)}|} \longrightarrow 0 \quad \text{as } k \to \infty,$$

so q_k should be smaller than q_* when k is sufficiently large. Therefore, when $q_k > q_*$, linear convergence seems apparent. In this case, for sufficiently large k,

$$\frac{|x^{(k+1)} - x^*|}{|x^{(k)} - x^*|} \approx q_{n,M} \quad \text{or} \quad |x^{(k+1)} - x^*| \approx q|x^{(k)} - x^*|, \qquad \text{where } q \equiv q_{n,M},$$

```
     Input:  subscript $i$, initial end points $\hat{\lambda}_i, \hat{\lambda}_{i+1}$.
     Output: starting iterates $x^{(0)}, x^{(1)}$ and $\kappa(x^{(0)}), \kappa(x^{(1)}), -\frac{f'(x^{(0)})}{f(x^{(0)})}, -\frac{f'(x^{(1)})}{f(x^{(1)})}$.
     Begin IniPts
         $(a,b) = (\hat{\lambda}_i, \hat{\lambda}_{i+1})$; Evaluate $-\frac{f'(a)}{f(a)}$, $\kappa(a)$, $-\frac{f'(b)}{f(b)}$, $\kappa(b)$ by DetEvl;
(##)     If $-\frac{f'(a)}{f(a)} < 0$ and $-\frac{f'(b)}{f(b)} > 0$ then
             $c = \frac{a+b}{2}$; Evaluate $-\frac{f'(c)}{f(c)}$, $\kappa(c)$ by DetEvl;
             $(a,b) = \begin{cases} (c,b), & \text{if } c < \lambda_i \text{ ( i.e. } \kappa(c) < i \text{)} \\ (a,c), & \text{if } c > \lambda_i \text{ ( i.e. } \kappa(c) \geq i \text{)} \end{cases}$ ;   Goto (##);
         else if $-\frac{f'(a)}{f(a)} < 0$ and $-\frac{f'(b)}{f(b)} < 0$ then
             $c = \max\left\{b - \frac{f'(b)}{f(b)}, \frac{a+b}{2}\right\}$; Evaluate $-\frac{f'(c)}{f(c)}$, $\kappa(c)$ by DetEvl;
             If $c > \lambda_i$ then $(x^{(0)}, x^{(1)}) = (b,c)$ ; Goto (#) ;
                 else $(a,b) = (c,b)$; Goto (##) ;
         else if $-\frac{f'(a)}{f(a)} > 0$ and $-\frac{f'(b)}{f(b)} > 0$ then
             $c = \min\left\{a - \frac{f'(a)}{f(a)}, \frac{a+b}{2}\right\}$; Evaluate $-\frac{f'(c)}{f(c)}$, $\kappa(c)$ by DetEvl;
             If $(c < \lambda_i)$ then $(x^{(0)}, x^{(1)}) = (a,c)$; Goto (#) ;
                 else $(a,b) = (a,c)$; Goto (##) ;
         else
             $c = \begin{cases} \min\left\{a - \frac{f'(a)}{f(a)}, \frac{a+b}{2}\right\}, & \text{if } \left|\frac{f'(a)}{f(a)}\right| \leq \left|\frac{f'(b)}{f(b)}\right| \\ \max\left\{b - \frac{f'(b)}{f(b)}, \frac{a+b}{2}\right\}, & \text{otherwise} \end{cases}$
             Evaluate $-\frac{f'(c)}{f(c)}$, $\kappa(c)$ by DetEvl;
             set $(x^{(0)}, x^{(1)}) = \begin{cases} (a,c), & \text{if } c < \lambda_i \\ (b,c), & \text{if } c > \lambda_i \end{cases}$
         End if ;
(#)  End IniPts
```

FIG. 4. *Algorithm* IniPts.

and

$$q_k = \frac{|x^{(k+1)} - x^{(k)}|}{|x^{(k)} - x^{(k-1)}|} = \frac{|x^{(k)} - x^*| - |x^{(k+1)} - x^*|}{|x^{(k-1)} - x^*| - |x^{(k)} - x^*|} \approx \frac{(1-q)|x^{(k)} - x^*|}{(1-q)|x^{(k-1)} - x^*|} \approx q.$$

In practice, q_k's can be very close to q after several iterations. If $q_j = q$ for $j \geq k$, then

$$|x^{(j+1)} - x^*| = q|x^{(j)} - x^*| \quad \text{for } j \geq k$$

and

$$x^* = x^{(k)} + \frac{1}{1-q}(x^{(k+1)} - x^{(k)}).$$

So, at the kth iterate $x^{(k)}$, if linear convergence is revealed, namely $q_k > q_*$, then instead of performing the regular quasi-Laguerre iteration

$$x^{(k+1)} = x^{(k)} + \delta_k,$$

where δ_k is the correction $x^{(k+1)} - x^{(k)}$ from $x^{(k)}$ calculated by the quasi-Laguerre iteration with $m = 1$, one can accelerate the iteration by setting

$$x^{(k+1)} = x^{(k)} + \frac{\delta_k}{1 - q_k}.$$

```
Input:  subscript $i$, starting iterates $x^{(0)}$, $x^{(1)}$ with $\kappa(x^{(0)})$,
        $\kappa(x^{(1)})$, $-\frac{f'(x^{(0)})}{f(x^{(0)})}$ and $-\frac{f'(x^{(1)})}{f(x^{(1)})}$ evaluated by INIPTS.
output: the $i$-th eigenvalue $\lambda_i$ of $T$.
Define $q_* = q_{3,2}$: the positive solution of $x^2+x-0.5=0$
                 (the smallest linear ratio for multiple zeros);
Begin AccQ-LAG
    For $k = 1, 2, \cdots$
        $c = \begin{cases} u_+, & \text{if } x^{(k)} < \lambda_i \\ u_-, & \text{if } x^{(k)} > \lambda_i \end{cases}$ ($u_\pm$ are from quasi-Lag.formulae);
        $\delta_k = c - x^{(k)}$; $q = \frac{\delta_k}{\delta_{k-1}}$; $tol = 2.5\varepsilon\left[\max_j(|\beta_j| + |\beta_{j+1}|)\right] + |x^{(k)}|\varepsilon$ ;
        If ($|\delta_k| < tol$) or ($k > 2$ and $\frac{\delta_k^2}{|\delta_{k-1}|-|\delta_k|} < tol$) Goto (#) ;
        If ($q > q_*$) then
            $c = x^{(k)} + \frac{\delta_k}{1-q}$; evaluate $-\dfrac{f'(c)}{f(c)}$ and $\kappa(c)$ by DETEVL;
            If ($c$ is on the same side of $\lambda_i$ as $x^{(k)}, x^{(k-1)}$) then
                Goto (##);
            else, For $l = 8, 4, 2, 1$,
                    $c = x^{(k)} + \dfrac{\delta_k(1-q^l)}{1-q}$;
                    evaluate the Sturm sequence $\kappa(c)$ ;
                    If ($c$ is on the same side of $\lambda_i$ as $x^{(k)}, x^{(k-1)}$)
                        Goto (##);
                  End for
        End if
(##)    $x^{(k+1)} = c$ ;
    End for
(#) End AccQ-LAG
```

FIG. 5. *Algorithm* ACCQ-LAG.

The fundamental features of the acceleration process described in this section are summarized in the algorithm ACCQ-LAG (Figure 5). Table 1 gives the comparison between the original quasi-Laguerre iteration and the accelerated quasi-Laguerre iteration. The Wilkinson matrix W_{99}^+ is used here with the starting points $x^{(0)} = 11.5$ and $x^{(1)} = L_-(x^{(0)}) = 11.27$ targeting $\lambda_{23} = 11.0 + 5 \times 10^{-15}$ of W_{99}^+, while $\lambda_{22} = 11.0 - 5 \times 10^{-15}$. Also note in the third column of Table 1 that the q_k's are identical up to six digits with $q = q_{99,2} \approx 0.402522$.

6. Numerical tests. Our algorithm was implemented and tested on SPARC stations with IEEE floating point standard. The machine precision was $\varepsilon \approx 2.2 \times 10^{-16}$.

6.1. Testing matrices. There are 12 types of matrices used in testing our algorithm. In the following description of these matrix types, $\alpha_i, i = 1, \ldots, n$, denote the diagonal entries and $\beta_i, i = 1, \ldots, n-1$, are the subdiagonal entries.

6.1.1. Matrices with known eigenvalues.

Type 1. Toeplitz matrices $[b, a, b]$. Exact eigenvalues: $\{a + 2b\cos\frac{k\pi}{n+1}\}_{1\le k\le n}$ [7, Ex. 7.4, p. 137].

Type 2. $\alpha_1 = a - b, \alpha_i = a$ for $i = 2, \ldots, n-1, \alpha_n = a + b$. $\beta_j = b, j = 1, \ldots, n-1$. Exact eigenvalues: $\{a + 2b\cos\frac{(2k-1)\pi}{2n}\}_{1\le k\le n}$ [7, Ex. 7.6, p. 138].

Type 3. $\alpha_i = \begin{cases} a & \text{for odd } i, \\ b & \text{for even } i, \end{cases}$ $\beta_i = 1$. Exact eigenvalues :

TABLE 1

Acceleration of the quasi-Laguerre iteration for W_{99}^{+}. For each number in the table, the digits before a space are correct.

k	$x^{(k)}$ (original)	$q_k = \frac{\lvert x^{(k+1)} - x^{(k)}\rvert}{\lvert x^{(k)} - x^{(k-1)}\rvert}$	$x^{(k)}$ (accelerated)
1	11 .500000000000000		11. 500000000000000
2	11 .270700712327294		11. 270700712327294
3	11 .149428542073966	0. 706618	11.0 13287361699255
4	11.0 63735435125402	0.4 37604	11.000 124299930121
5	11.0 26235830973979	0.4 17533	11.0000 61856842477
6	11.0 10578521008206	0.40 3427	11.000000 198368395
7	11.00 4261938886133	0.40 3112	11.0000000 99023529
8	11.00 1715645778917	0.4025 55	11.0000000000 12077
9	11.000 690622605815	0.4025 56	11.00000000000 6040
10	11.000 277993423158	0.40252 7	11.00000000000 2304
11	11.000 111899071095	0.40252 5	11.000000000000 156
12	11.0000 45041905516	0.40252 3	11.0000000000000 64
13	11.0000 18130368451	0.402522	
14	11.00000 7297872838	0.402522	
15	11.00000 2937554150	0.402522	
16	11.00000 1182429996	0.402522	
17	11.000000 475954003	0.402522	
18	11.000000 191581922	0.402522	
19	11.0000000 77115924	0.402522	
20	11.0000000 31040850	0.402522	
21	11.0000000 12494622	0.402522	
22	11.00000000 5029360	0.402522	
23	11.00000000 2024429	0.402522	
24	11.000000000 814877	0.402522	
25	11.000000000 328006	0.402522	
26	11.000000000 132029	0.402520	
27	11.0000000000 53145	0.40252 1	
28	11.0000000000 21393	0.40252 6	
29	11.00000000000 8612	0.40252 1	
30	11.00000000000 3467	0.402 477	
31	11.00000000000 1396	0.40 1680	
32	11.000000000000 565	0. 399604	
33	11.000000000000 233	0. 381937	
34	11.000000000000 107	0. 305425	

$$\left\{ \frac{a + b \pm \sqrt{(a-b)^2 + 16\cos^2 \frac{k\pi}{n+1}}}{2} \right\}_{1 \le k \le n/2} \qquad \text{(add } \{a\} \text{ when } n \text{ is odd)}$$

[7, Ex. 7.8 and 7.9, p. 139].

Type 4. $\alpha_i = 0$, $\beta_i = \sqrt{i(n-i)}$. Exact eigenvalues: $\{-n + 2k - 1\}_{1 \le k \le n}$ [7, Ex. 7.10, p. 140].

Type 5. $\alpha_i = -[(2i-1)(n-1) - 2(i-1)^2]$, $\beta_i = i(n-i)$. Exact eigenvalues: $\{-k(k-1)\}_{1 \le k \le n}$ [7, Ex. 7.11, p. 141].

6.1.2. Wilkinson and random matrices.

Type 6. Wilkinson matrices W_n^{+}. $\beta_i = 1$,

$$\alpha_i = \begin{cases} n/2 - i + 1 & \text{for even } n \text{ and } 1 \le i \le n/2, \\ i - n/2 & \text{for even } n \text{ and } n/2 < i \le n, \\ (n-1)/2 - i + 1 & \text{for odd } n \text{ and } 1 \le i \le (n+1)/2, \\ i - (n+1)/2 & \text{for odd } n \text{ and } (n+1)/2 < i \le n, \end{cases}$$

[16, pp. 308–309]. Most of the eigenvalues are in pairs, consisting of two numerically indistinguishable eigenvalues.

Type 7. Random matrices. The α_i's and β_i's are random numbers in [0, 1].

6.1.3. LAPACK testing matrices. (These are generated by the LAPACK test matrix generator [1].)

Type 8. Matrices with eigenvalues evenly distributed between their smallest and largest eigenvalues.

Type 9. Matrices with geometrically distributed eigenvalues. Namely, eigenvalues can be written as $\{q^k\}_{1\le k\le n}$ for some $q \in (0, 1)$.

Type 10. Matrices with an eigenvalue 1 and the remaining eigenvalues in $(-\varepsilon, \varepsilon)$.

Type 11. Matrices with eigenvalues evenly distributed in the interval (0, 1] except one eigenvalue with very small magnitude.

Type 12. Matrices with an eigenvalue 1 and the rest of the eigenvalues are evenly distributed in a small interval $[10^{-12} - \varepsilon, 10^{-12} + \varepsilon]$.

6.2. Speed test in evaluating eigenvalues without computing eigenvectors. We compare the performance of the following codes for evaluating eigenvalues of an $n \times n$ matrix:

(1) Q-LAG, our split–merge algorithm using the quasi-Laguerre iteration. Storage requirement, $9n$;

(2) LAG, split–merge algorithm using Laguerre's iteration developed by Li and Zeng [10]. Storage requirement, $9n$;

(3) B/M, bisection/multisection subroutine DSTEBZ in LAPACK. Storage requirement, $12n$;

(4) RFQR, root-free QR routine DSTERF in LAPACK, as recommended in LAPACK for evaluating eigenvalues only. Storage requirement, $2n$.

The results of this test are given in Figure 6. Matrices of Types 9–12 involve tiny eigenvalues or clusters and are used more in stability tests (See §6.4). The speed comparison in this category may not be relevant. In particular, matrices of Types 10 and 12 have big dense clusters, and intensive deflations make Q-LAG as well as LAG outscore RFQR in speed by a wide margin.

For all other matrices (Types 1–8), Q-LAG is much faster than LAG (especially for Types 1, 4, and 5) and LAG is much faster than B/M. RFQR is apparently the fastest code in almost all cases, except for matrices of Types 6 and 7 where Q-LAG is about 70% and 40% ($n = 500$) faster, respectively. Overall, Q-LAG performs very similarly to RFQR and is much more competitive than B/M. Taking into account Q-LAG's flexibility in evaluating the partial spectrum and the potential in parallel computing, Q-LAG is by far the most efficient algorithm among all the algorithms being tested here.

6.3. Speed test in evaluating eigenpairs. We compare the performance of the following codes for evaluating the eigenpairs of an $n \times n$ matrix:

(1) Q-LAG, our split–merge algorithm using the quasi-Laguerre iteration plus the inverse iteration subroutine DSTEIN in LAPACK. Storage requirement, $n^2 + O(n)$;

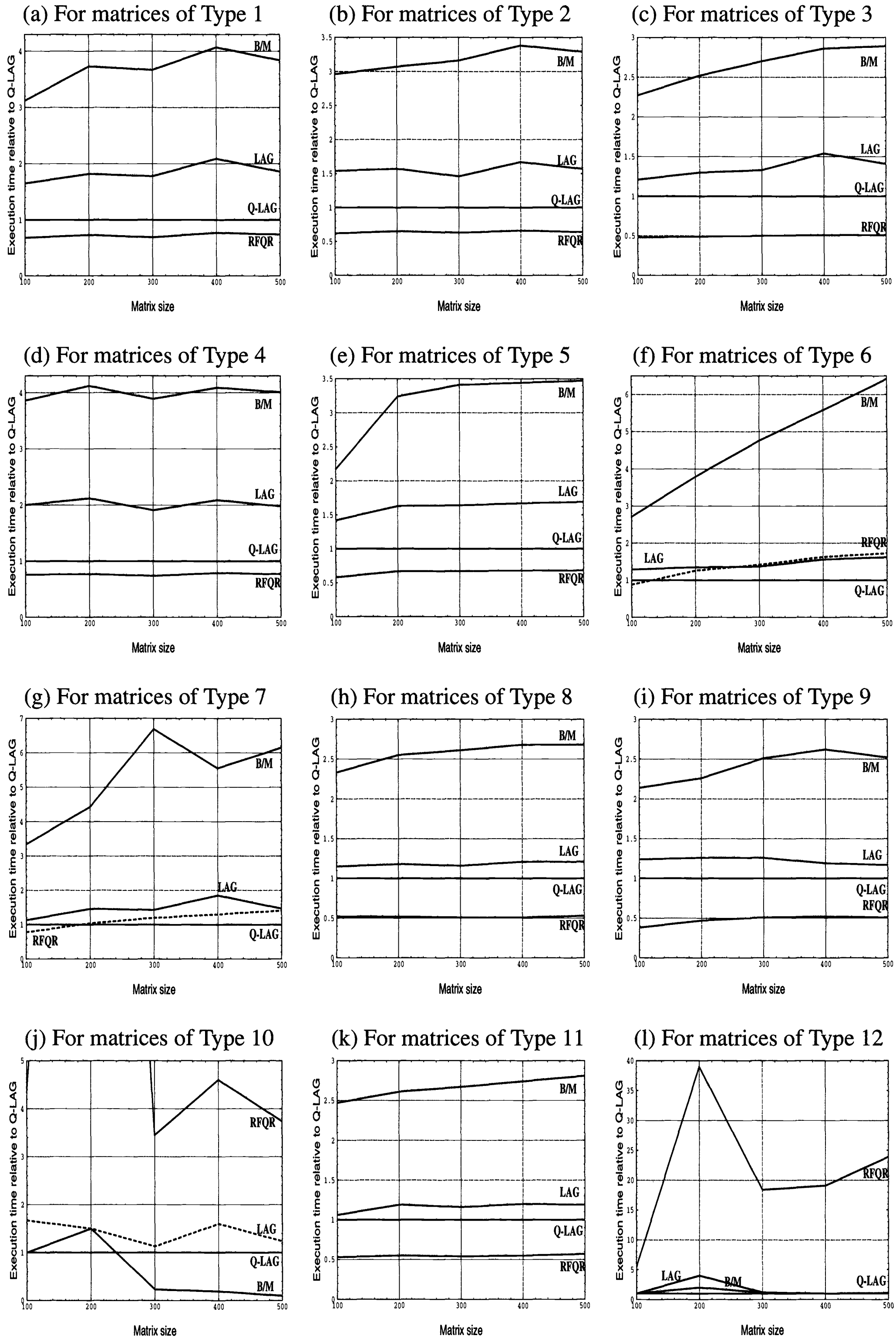
(2) B/M, DSTEBZ plus DSTEIN. Storage requirement, $n^2 + O(n)$;

(3) INV, DSTEIN with eigenvalues computed by Q-LAG (for analysis purposes);

(4) D&C, divide-and-conquer subroutine DSTEDC in LAPACK. Storage requirement, $3n^2 + O(n)$;

(5) QR, QR routine DSTEQR in LAPACK, as recommended in LAPACK for evaluating eigenpairs only. Storage requirement, $n^2 + O(n)$.

The storage requirement for D&C is relatively higher than those which use the inverse iteration code. The results of evaluations of all eigenpairs and one-third of the largest eigenvalues with corresponding eigenvectors are listed in columns A and B, respectively, in Table 2.



(a) For matrices of Type 1

(b) For matrices of Type 2

(c) For matrices of Type 3

(d) For matrices of Type 4

(e) For matrices of Type 5

(f) For matrices of Type 6

(g) For matrices of Type 7

(h) For matrices of Type 8

(i) For matrices of Type 9

(j) For matrices of Type 10

(k) For matrices of Type 11

(l) For matrices of Type 12

FIG. 6. *Relative execution time for evaluating all eigenvalues without computing eigenvectors. B/M, DSTEBZ; LAG, Laguerre's iteration; Q-LAG, the quasi-Laguerre iteration; RFQR, DSTERF.*

TABLE 2

Speed test on all matrices (for Types 1–3, we choose $a = 2, b = 1$). The execution time in seconds for evaluating all eigenpairs is listed in column A. The execution time in seconds for evaluating only one-third of the largest eigenpairs is listed in column B (only Q-LAG and B/M have this feature).

Type		$n = 100$		$n = 200$		$n = 300$		$n = 400$		$n = 500$	
		A	B	A	B	A	B	A	B	A	B
1	Q-LAG	1.15	0.42	4.31	1.49	9.95	3.42	18.1	6.49	31.1	11.2
	B/M	1.69	0.60	6.84	2.42	14.9	5.17	27.7	9.57	43.9	15.8
	INV	0.90	0.32	3.49	1.16	8.14	2.76	15.2	5.44	26.4	9.60
	D&C	0.54	—	3.18	—	9.36	—	21.0	—	40.4	—
	QR	1.89	—	13.8		44.0	—	102.	—	196.	—
2	Q-LAG	1.18	0.43	4.44	1.57	10.3	3.60	19.0	6.55	31.9	11.6
	B/M	1.73	0.60	6.50	2.36	14.9	5.22	27.4	9.60	44.1	16.1
	INV	0.92	0.31	3.50	1.22	8.17	2.84	15.6	5.31	26.4	9.58
	D&C	0.76	—	4.62	—	14.0	—	31.3	—	59.6	—
	QR	1.85	—	13.9	—	45.0	—	104.	—	198.	—
3	Q-LAG	1.26	0.42	4.74	1.61	10.7	3.67	19.8	6.79	33.0	11.8
	B/M	1.67	0.62	6.60	2.34	15.1	5.14	27.0	0.68	44.8	16.1
	INV	0.93	0.31	3.56	1.21	8.21	2.81	15.8	5.47	26.8	9.78
	D&C	0.77	—	4.92	—	14.8	—	33.6	—	63.6	—
	QR	1.83	—	13.1	—	40.8	—	93.4	—	179.	—
4	Q-LAG	1.13	0.41	4.29	1.60	9.57	3.20	16.9	5.69	26.5	8.95
	B/M	1.69	0.62	6.69	2.33	14.7	5.15	26.0	9.14	40.4	14.1
	INV	0.92	0.32	3.54	1.20	7.83	2.63	14.0	4.77	21.9	7.34
	D&C	0.51	—	3.18	—	9.32	—	21.6	—	39.8	—
	QR	1.78	—	13.9	—	45.2	—	105.	—	203.	—
5	Q-LAG	1.16	0.43	4.57	1.67	10.8	4.35	21.5	10.2	40.2	22.1
	B/M	1.67	0.58	6.56	2.38	15.6	5.93	29.8	13.2	52.7	26.6
	INV	0.90	0.32	3.66	1.31	8.89	3.60	18.1	8.93	35.0	20.2
	D&C	0.55	—	3.23	—	9.31	—	21.2	—	39.1	—
	QR	1.74	—	13.7	—	44.3	—	103.	—	197.	—
6	Q-LAG	1.10	0.37	4.02	1.33	8.73	2.86	15.2	5.07	23.5	7.76
	B/M	1.48	0.50	5.23	1.77	11.7	4.11	20.4	6.93	31.9	10.8
	INV	0.93	0.32	3.59	1.21	7.87	2.63	14.0	4.71	22.0	7.31
	D&C	0.29	—	0.79	—	1.37	—	2.05	—	2.77	—
	QR	1.73	—	12.5	—	41.8	—	98.8	—	183.	—
7	Q-LAG	1.14	0.39	4.33	1.49	9.43	3.25	16.6	5.63	25.8	8.67
	B/M	1.70	0.67	6.68	2.34	14.9	5.20	26.1	9.19	40.9	14.2
	INV	0.91	0.31	3.66	1.25	8.15	2.78	14.5	4.90	22.9	7.65
	D&C	0.70	—	2.13	—	4.09	—	7.29	—	10.5	—
	QR	2.01	—	16.3	—	53.7	—	123.	—	234.	—
8	Q-LAG	1.24	0.44	4.79	1.69	10.6	3.58	18.8	6.36	29.1	9.85
	B/M	1.71	0.66	6.67	2.41	14.9	5.14	26.0	9.08	40.5	14.1
	INV	0.91	0.31	3.60	1.25	8.08	2.71	14.4	4.87	22.3	7.62
	D&C	0.83	—	4.95	—	14.7	—	32.8	—	59.6	—
	QR	1.86	—	14.4	—	45.0	—	104.	—	200.	—
9	Q-LAG	2.21	0.54	14.0	2.40	41.5	6.34	92.5	12.7	181.	24.3
	B/M	2.48	0.65	14.9	3.01	43.6	7.51	96.4	15.0	187.	27.7
	INV	2.00	0.44	13.2	2.13	40.1	5.77	90.0	11.8	177.	22.7
	D&C	0.36	—	1.69	—	4.19	—	8.95	—	15.3	—
	QR	1.49	—	11.2	—	35.3	—	79.1	—	157.	—
10	Q-LAG	2.71	0.51	0.24	0.12	2.82	1.16	132.	18.4	28.4	8.68
	B/M	2.57	0.52	16.8	2.76	52.4	7.83	120.	17.3	229.	30.6
	INV	2.68	0.48	0.22	0.09	2.55	0.98	131.	18.1	27.7	8.38
	D&C	0.55	—	0.49	—	0.75	—	1.38	—	1.69	—
	QR	1.56	—	11.8	—	37.0	—	87.4	—	172.	—
11	Q-LAG	1.24	0.42	4.78	1.66	10.6	3.59	18.5	6.36	28.8	9.72
	B/M	1.72	0.65	6.60	2.32	14.8	5.09	26.0	9.05	40.2	14.1
	INV	0.92	0.31	3.63	1.25	8.12	2.74	14.2	5.86	22.4	7.52
	D&C	0.81	—	4.90	—	14.5	—	32.6	—	59.5	—
	QR	1.85	—	14.7	—	45.4	—	105.	—	205.	—
12	Q-LAG	0.13	0.13	0.07	0.07	1.06	0.98	7.45	6.81	0.18	0.17
	B/M	2.51	0.54	16.9	2.74	53.0	8.07	120.	17.2	229.	31.1
	INV	0.12	0.11	0.07	0.04	1.04	0.93	7.42	6.75	0.17	0.10
	D&C	0.23	—	0.52	—	0.79	—	1.51	—	1.75	—
	QR	1.38	—	9.36	—	31.1	—	67.9	—	138.	—

TABLE 3

Accuracy, relative to machine precision ε, on all matrices (for Types 1–3, we choose $a = 2, b = 1$). Note that only the first five types of matrices have valid direct error $\mathcal{D}$.

Type		$n = 100$			$n = 200$			$n = 300$			$n = 400$			$n = 500$		
		$\mathcal{R}$	$\mathcal{O}$	$\mathcal{D}$	$\mathcal{R}$	$\mathcal{O}$	$\mathcal{D}$	$\mathcal{R}$	$\mathcal{O}$	$\mathcal{D}$	$\mathcal{R}$	$\mathcal{O}$	$\mathcal{D}$	$\mathcal{R}$	$\mathcal{O}$	$\mathcal{D}$
1	Q-LAG	0.84	76.1	0.50	1.27	189.	1.00	1.41	152.	1.00	1.89	186.	1.00	2.16	158.	1.00
	B/M	0.84	75.7	1.00	1.27	154.	1.00	1.49	143.	1.00	1.85	145.	1.00	1.96	173.	1.00
	D&C	3.60	11.5	1.50	3.52	13.9	2.00	3.12	16.6	2.00	3.53	21.6	2.00	3.38	20.3	2.00
	RFQR	2.97	35.4	2.50	3.72	140.	4.00	2.97	121.	3.00	3.39	161.	3.00	6.27	146.	6.50
	QR	6.01	17.4	2.00	8.05	24.5	2.75	10.2	29.6	4.25	11.2	35.0	5.50	12.8	38.7	3.00
2	Q-LAG	0.71	66.7	1.00	1.20	156.	1.00	1.50	144.	1.00	1.61	191.	1.00	2.08	168.	1.00
	B/M	0.85	130.	1.00	1.16	166.	1.00	1.49	161.	1.50	1.76	189.	1.00	2.27	171.	1.50
	D&C	3.60	10.7	1.50	3.71	14.0	1.50	3.38	17.3	2.00	3.81	19.7	2.00	3.42	22.9	2.00
	RFQR	2.35	62.5	2.50	3.52	157.	3.50	3.27	177.	3.00	3.01	80.5	3.00	3.71	149.	4.00
	QR	6.15	19.0	2.25	8.46	25.6	3.00	10.3	28.4	4.06	12.0	37.9	3.69	13.0	47.1	6.13
3	Q-LAG	0.70	72.5	0.50	1.02	72.1	0.50	1.27	112.	0.50	1.50	137.	0.50	1.78	111.	0.50
	B/M	1.13	77.9	1.00	1.13	76.3	1.00	1.16	77.1	1.00	1.39	109.	1.00	1.69	160.	1.00
	D&C	3.67	12.1	1.00	3.80	14.8	1.50	3.50	18.5	1.13	3.99	20.9	2.00	3.93	24.2	1.00
	RFQR	2.21	64.5	2.00	2.65	99.3	2.50	3.29	64.7	3.00	2.79	90.4	2.50	4.23	108.	2.50
	QR	6.57	20.1	2.00	9.06	26.9	2.63	10.7	26.5	3.75	12.7	36.0	3.19	13.9	38.2	4.00
4	Q-LAG	0.71	11.3	0.16	0.83	14.4	0.08	1.81	51.2	0.85	1.44	47.7	0.09	1.13	48.0	0.51
	B/M	0.92	12.1	0.64	1.60	45.0	1.28	1.81	72.3	0.85	1.44	56.3	1.28	2.14	121.	1.02
	D&C	5.22	11.4	2.56	6.33	12.8	2.56	6.61	15.6	1.71	7.64	16.2	3.20	8.81	19.6	2.05
	RFQR	11.0	13.1	10.9	18.0	21.5	17.9	26.9	63.1	27.3	20.6	49.5	20.5	23.1	60.4	13.3
	QR	14.9	19.1	10.2	23.9	31.6	17.9	27.0	35.0	8.53	30.7	37.5	14.1	32.2	46.4	14.8
5	Q-LAG	0.48	21.1	0.10	0.53	19.1	0.05	0.66	23.9	0.52	0.68	18.8	0.52	0.97	32.5	0.52
	B/M	0.89	24.5	0.82	0.89	54.2	0.82	0.84	68.3	1.05	0.91	40.4	1.05	1.08	56.5	1.05
	D&C	4.05	12.2	1.64	3.78	16.9	1.64	2.98	13.6	3.15	3.94	17.5	3.15	3.22	16.1	3.15
	RFQR	2.91	28.1	2.87	4.94	29.0	4.92	6.61	38.8	5.76	11.5	42.0	5.76	5.76	44.8	5.76
	QR	9.06	24.7	5.74	11.1	25.8	6.55	10.8	29.6	6.29	14.4	35.9	6.29	14.6	47.3	6.29
6	Q-LAG	0.31	3.54	—	0.28	5.57	—	0.38	4.03	—	0.28	5.55	—	1.02	4.53	—
	B/M	1.06	4.05	—	0.92	5.57	—	0.85	4.53	—	1.36	5.56	—	1.02	4.55	—
	D&C	2.29	16.7	—	3.22	19.6	—	3.06	15.9	—	2.07	20.2	—	3.01	10.2	—
	RFQR	6.56	7.01	—	13.3	6.02	—	18.7	4.54	—	30.6	5.54	—	19.9	4.05	—
	QR	12.5	74.5	—	9.06	224.	—	16.3	360.	—	14.3	486.	—	12.1	632.	—
7	Q-LAG	0.77	4.85	—	0.92	10.8	—	1.56	30.0	—	1.49	30.8	—	1.43	18.7	—
	B/M	1.13	14.8	—	0.85	51.6	—	0.99	30.0	—	1.16	30.9	—	1.07	18.4	—
	D&C	7.29	26.2	—	6.23	26.7	—	5.70	13.2	—	10.0	22.3	—	7.58	13.4	—
	RFQR	10.6	5.33	—	12.3	23.8	—	24.3	14.4	—	31.1	66.6	—	24.3	23.4	—
	QR	12.0	84.4	—	20.1	180.	—	25.3	249.	—	31.0	253.	—	31.7	343.	—
8	Q-LAG	0.84	20.8	—	0.92	37.9	—	1.17	78.9	—	1.45	112.	—	1.39	127.	—
	B/M	0.92	14.6	—	0.96	37.1	—	1.36	88.1	—	1.61	104.	—	1.19	109.	—
	D&C	8.11	13.7	—	6.60	16.2	—	7.90	21.1	—	7.79	20.0	—	8.30	23.5	—
	RFQR	4.66	13.5	—	6.56	40.4	—	7.30	57.3	—	6.20	82.2	—	28.9	55.8	—
	QR	12.0	24.1	—	16.4	27.6	—	20.9	42.1	—	22.3	36.9	—	37.3	71.3	—
9	Q-LAG	1.29	9.98	—	7.44	26.9	—	24.5	5.75	—	5.44	15.4	—	97.4	45.8	—
	B/M	0.69	4.18	—	1.25	5.28	—	2.15	5.64	—	10.7	47.6	—	20.5	15.6	—
	D&C	4.26	18.2	—	6.76	30.1	—	3.94	32.3	—	5.61	28.1	—	4.74	18.0	—
	RFQR	4.46	3.56	—	4.97	7.10	—	7.13	11.2	—	9.17	9.94	—	8.10	10.9	—
	QR	4.72	42.3	—	4.82	80.6	—	8.41	123.	—	10.7	157.	—	10.6	206.	—
10	Q-LAG	1.50	213.	—	0.91	3.14	—	2.11	39.2	—	2.99	9607	—	3.33	628.	—
	B/M	0.58	257.	—	1.40	45.4	—	1.28	1800	—	2.01	484.	—	2.88	5683	—
	D&C	1.14	19.4	—	1.31	16.2	—	1.36	13.5	—	1.41	18.0	—	1.49	15.3	—
	RFQR	0.01	4.91	—	0.22	9.09	—	0.08	10.7	—	0.17	11.4	—	0.02	13.4	—
	QR	0.03	42.3	—	0.41	67.0	—	0.58	95.5	—	0.17	194.	—	0.49	210.	—
11	Q-LAG	0.45	9.57	—	0.98	28.9	—	0.60	42.5	—	0.69	83.4	—	0.83	61.8	—
	B/M	0.81	9.64	—	1.02	40.6	—	0.98	40.0	—	1.00	68.7	—	0.91	71.3	—
	D&C	4.06	13.8	—	3.62	16.5	—	10.9	23.0	—	3.29	18.1	—	4.48	24.8	—
	RFQR	8.48	11.7	—	3.09	25.8	—	10.9	23.0	—	14.1	54.9	—	16.7	35.0	—
	QR	8.63	19.7	—	9.44	29.7	—	14.5	47.6	—	18.0	53.8	—	18.0	81.7	—
12	Q-LAG	1.74	4.73	—	1.75	2.17	—	1.11	13.7	—	1.75	135.	—	3.16	3.38	—
	B/M	1.23	102.	—	6.48	3496	—	1.48	18.5	—	0.70	92.6	—	0.80	43.0	—
	D&C	0.37	17.2	—	0.28	17.4	—	0.52	14.0	—	1.01	20.0	—	1.53	11.9	—
	RFQR	0.23	5.25	—	0.28	10.2	—	0.48	12.3	—	0.02	16.1	—	0.06	14.8	—
	QR	0.45	38.6	—	0.64	80.8	—	0.49	148.	—	0.04	228.	—	0.64	358.	—

In comparing the execution time in evaluating all eigenpairs, our algorithm leads QR substantially. For matrices of Types 6, 9, and 10, where a number of clusters exist, our algorithm is slower than D&C due to drastic deflations of clusters. But this sort of matrix scarcely occurs in practice; it is often used for testing the robustness of an algorithm. For all other types of matrices, our algorithm leads D&C for large matrices ($n \geq 300$), while for small matrices, D&C is faster since it takes much longer for INV to compute eigenvectors in our algorithm.

Only Q-LAG, LAG, and B/M can be applied efficiently to evaluate one-third of the largest eigenvalues and their corresponding eigenvectors. D&C and QR must evaluate all the eigenpairs to obtain the largest one-third. This makes our algorithm more flexible and efficient. Compared with B/M in this category, our algorithm leads clearly for all types of matrices.

6.4. Accuracy test. For those matrices with known eigenvalues (Types 1–5), the accuracy of a method can be determined by

$$\text{direct error: } \mathcal{D} = \max_i \frac{|\tilde{\lambda}_i - \lambda_i|}{\|T\|_1},$$

where $\tilde{\lambda}_i$ is the approximation of the exact eigenvalue λ_i of T and $\|\cdot\|_1$ is the l_1-norm. For matrices of other types, the exact eigenvalues are unavailable; the accuracy of a method is thus determined by the residual error and the orthogonality error in the computed solution. For $T = QDQ^T$ with computed solution $\tilde{Q}\tilde{D}\tilde{Q}^T$, these errors are computed by

$$\text{residual error: } \mathcal{R} = \max_i \frac{\|(T\tilde{Q} - \tilde{Q}\tilde{D})e_i\|_2}{|\tilde{\lambda}|_{\max}},$$

$$\text{orthogonality error: } \mathcal{O} = \max_i \|(\tilde{Q}^T\tilde{Q} - I)e_i\|_2.$$

If the residual and orthogonality error are small, then the error matrix $T - \tilde{Q}\tilde{D}\tilde{Q}^T$ is small [11].

The results are listed in Table 3. It appears that our algorithm Q-LAG achieves the smallest direct error on all matrices of the first five types. The direct error of our algorithm as well as B/M is independent of the matrix size, whereas RFQR and QR seem to have larger direct error when the matrix size becomes larger. For nearly all matrices, the eigenvectors generated by D&C suffer the least loss of orthogonality, whereas our algorithm and B/M enjoy the smallest residual errors, except on those occasions where clusters of eigenvalues exist and the inverse iteration fails to compute orthogonal vectors.

REFERENCES

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorenson, *LAPACK User's Guide*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.

[2] J. J. M. Cuppen, *A divide and conquer method for the symmetric tridiagonal eigenproblem*, Numer. Math., 36 (1981), pp. 177–195.

[3] Q. Du, M. Jin, T. Y. Li, and Z. Zeng, *Quasi-Laguerre iteration*, Math. Comp., to appear.

[4] J. J. Dongarra and D. C. Sorensen, *A fully parallel algorithm for the symmetric eigenvalue problem*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 139–154.

[5] L. V. Foster, *Generalizations of Laguerre's method: Lower order methods*, preprint, Mathematics and Computer Science Department, San Jose State University, San Jose, CA, 1982.

[6] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 2nd Ed., The Johns Hopkins University Press, Baltimore, MD, 1989.

[7] R. T. Gregory and D. L. Karney, *A Collection of Matrices for Testing Computational Algorithms*, Robert E. Krieger Publishing Company, Huntington, NY, 1978.

[8] M. Gu and S. C. Eisenstat, *A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem* SIAM J. Matrix Anal. Appl., 16 (1995), pp. 172–191.

[9] W. KAHAN, *Notes On Laguerre's Iteration*, preprint, University of California, Berkeley, CA, 1992.

[10] T. Y. LI AND Z. ZENG, *Laguerre's iteration in solving the symmetric tridiagonal eigenproblem—revisited*, SIAM J. Sci. Comput., 15 (1994), pp. 1145–1173.

[11] J. RUTTER, *A serial implementation of Cuppen's divide and conquer algorithm for the symmetric eigenvalue problem*, LAPACK Working Note 69, 1994.

[12] B. PARLETT, *The use of a refined error bound when updating eigenvalues of tridiagonals*, Linear Algebra Appl., 68 (1985), pp. 179–219.

[13] B. N. PARLETT, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ, 1980.

[14] D. C. SORENSEN AND P. T. P. TANG, *On the orthogonality of eigenvectors computed by divide-and-conquer techniques*, SIAM. J. Numer. Anal., 28 (1991), pp. 1752–1775.

[15] C. TREFFTZ, C. C. HUANG, P. MCKINLEY, T. Y. LI, AND Z. ZENG, *A scalable eigenvalue solver for symmetric tridiagonal matrices*, Parallel Comput., 21 (1995), pp. 1213–1240.

[16] J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Oxford University Press, Oxford, 1965.

SIAM J. SCI. COMPUT.
Vol. 17, No. 6, pp. 1369–1394, November 1996

005

PRECONDITIONED ITERATIVE METHODS FOR UNSTEADY NON-NEWTONIAN FLOW BETWEEN ECCENTRICALLY ROTATING CYLINDERS*

D. RH. GWYNLLYW† AND T. N. PHILLIPS‡

Abstract. The flow of a non-Newtonian lubricant between eccentrically rotating cylinders is considered. The fluid model includes a viscosity law which is shear thinning and pressure thickening. The governing equations are discretized in space using spectral elements and in time by a time-splitting scheme. The system of linear equations generated at each time step is solved using the conjugate gradient method. Particular emphasis is given to the choice of efficient preconditioners for both constant and variable viscosity problems. Eigenvalue spectra are presented for the preconditioned systems for a range of realistic choices of the material and geometrical parameters.

Key words. non-Newtonian fluid, spectral element method, preconditioned conjugate gradient method, eccentrically rotating cylinders

AMS subject classifications. 65M70, 76M30, 65F10

1. Introduction. The journal bearing is an essential part of all internal combustion engines as a means of transferring energy from the piston connecting rods to the rotating crankshaft (Figure 1). The *big-end* journal bearing consists essentially of an inner cylinder (the journal), which is part of the crankshaft, and an outer cylinder (the bearing), which is at the end of the connecting rod. A lubricating film of oil separates the three-dimensional region between the cylinders which are, in general, eccentrically positioned. The *big-end* bearing is driven by a time-dependent load, whilst the journal's motion is approximately circular about O, the centre of the *main* bearing. The *main* journal bearing again consists of a journal being part of the crankshaft, whilst the outer cylinder is fixed to the engine casing. If O was fixed in time then the journal within the *main* bearing would rotate about its own axis. Both the main and the big-end bearings are subject to elastic deformation.

The modelling of the behaviour of the lubricant has many important and complicated features. These include incompressibility; viscoelasticity; changes in viscosity with shear rate, strain rate, pressure, and temperature; oil feed; boundary lubrication to incorporate extreme effects when the cylinders are almost touching; and cavitation. In addition the imposition of a time-dependent load means that the centre of the journal traces out a nontrivial locus in space.

In the present study we restrict ourselves to the two-dimensional flow of a non-Newtonian fluid under static loading conditions (fixed eccentricity). This model is important in its own right and presents a genuine and challenging test for numerical solution techniques. Significant boundary layers can develop in the flow at high eccentricity, even at modest rotational speeds of the journal. The accurate resolution of such boundary layers presents a major challenge to numerical methods. The extremely high number of degrees of freedom required by traditional finite element methods to resolve these layers has led to high-order methods such as spectral or spectral element methods being tried. The present study concentrates on the feasibility of solving the static loading problem for different eccentricities with particular attention given to the speed of computation. The motivation for studying this problem is that the dynamic loading problem will involve tens of thousands of such calculations.

In previous work on the statically loaded model [18], [19], [20] a single domain spectral method was used which employed a bipolar coordinate transformation to map the region be-

*Received by the editors July 8, 1994; accepted for publication (in revised form) June 7, 1995. This work is funded by Shell Research Ltd., Thornton, UK.

†Department of Mathematics, University of Wales, Aberystwyth SY23 3BZ, United Kingdom (deg@aber.ac.uk).
‡Department of Mathematics, University of Wales, Aberystwyth SY23 3BZ, United Kingdom (tnp@aber.ac.uk).

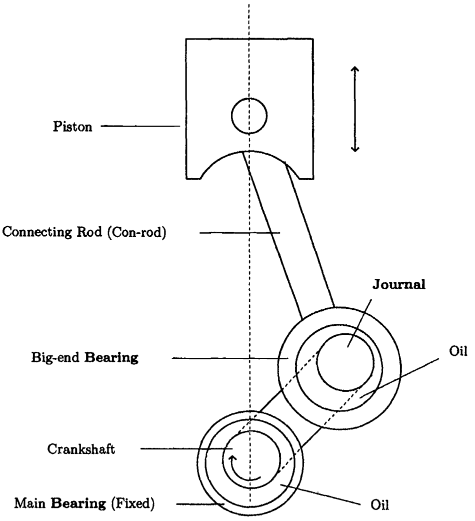


FIG. 1. *Schematic diagram showing the conversion of the translational motion of a piston to the rotational motion of the crankshaft via three journal bearing mechanisms.*

tween the journal and the bearing onto a rectangle. The flow variables were then approximated on this rectangle using Fourier–Chebyshev expansions. However, to allow for future possible deformation of the journal and bearing surfaces due to increased loads for the dynamically loaded problem we have decided to use a more versatile spectral element formulation.

In the spectral element method a variational formulation of the governing equations is used. This results in a symmetric positive definite system of linear equations to be solved at each time step. This system may be solved using a conjugate gradient iterative method. Improved efficiency is gained by constructing suitable preconditioners, thus reducing the number of iterations and the computational time to reach a converged solution. These are the issues that will be addressed in this paper.

2. Formulation of the problem. We consider the flow of a pressure-thickening generalized Newtonian fluid between eccentrically rotating cylinders. The inner (journal) and outer (bearing) cylinders are of radius R_J and R_B, respectively, with the distance between the centres of the cylinders given by e. The outer cylinder is kept at rest, while the inner cylinder is rotated at an angular velocity ω. The eccentricity is defined by $\epsilon = e/c$, where $c = R_B - R_J$ is known as the average gap. Let the region between the journal and the bearing be denoted by Ω, and let Γ_J and Γ_B denote the boundaries of the journal and bearing, respectively.

The governing equations for a generalized Newtonian fluid compose the conservation of momentum

$$\rho\left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v}.\nabla\mathbf{v}\right) = -\nabla p + \nabla.\mathbf{T}, \tag{1}$$

the conservation of mass

$$\nabla.\mathbf{v} = 0, \tag{2}$$

and the constitutive equation

$$\mathbf{T} = 2\eta(\dot\gamma, p)\mathbf{d}, \tag{3}$$

where ρ is the density, η is the variable viscosity, $\dot\gamma = \sqrt{2}\ \mathrm{tr}(\mathbf{d})^2$, $\mathbf{T}$ is the extra-stress tensor, and $\mathbf{d} = \frac{1}{2}(\nabla\mathbf{v} + (\nabla\mathbf{v})^T)$ is the rate of deformation tensor. Here tr$(\mathbf{A})$ denotes the trace of a tensor **A**. The constitutive equation written in this form can easily be modified to incorporate viscoelasticity [18]. The constitutive equation (3) is a modification of the usual generalized Newtonian model to include pressure dependence of the viscosity.

The viscosity law that we have used was proposed by Li and Davies [13]. It is shear thinning and pressure thickening. The various parameters in the model are determined empirically. The dependence of viscosity on $\dot{\gamma}$ and pressure is given by

$$\eta = \left\{ \eta_\infty + \frac{(\eta_0 - \eta_\infty)}{[1 + (K\dot{\gamma})^m]} \right\} \times \exp(-\alpha \mathrm{tr}(\sigma)/3 + F), \tag{4}$$

where K is a function of pressure

$$K = K(p) = \exp(-\bar{\alpha}\mathrm{tr}(\sigma)/3 + E),$$

$\sigma = -p\mathbf{I} + \mathbf{T}$ is the Cauchy stress tensor, and $\eta_0, \eta_\infty, m, n, \alpha, \bar{\alpha}$, E, and F are material parameters. This model describes the shear-thinning behaviour of the viscosity by a Cross-type formula [8]. Pressure thickening is modelled by a simple exponential law [1]. It is important to note that the viscosity law (4) is consistent with experiments [2] which span only limited ranges of the pressures which the lubricants experience under general operating conditions.

The governing equations (1)–(3) are solved subject to specified boundary and initial conditions. These are, respectively,

$$\mathbf{v} = \mathbf{v}_J \text{ on } \Gamma_J, \quad \mathbf{v} = \mathbf{0} \text{ on } \Gamma_B, \tag{5}$$

$$\mathbf{v}(\mathbf{x}, t = 0) = \mathbf{v}_0(\mathbf{x}). \tag{6}$$

3. Analysis of the steady Stokes problem. In this section we review previous work on the spectral discretization of the steady Stokes problem in order to motivate our choice of discrete velocity and pressure approximation spaces. For simplicity we consider the domain $D = [-1, 1] \times [-1, 1]$. The problem is to find a velocity $\mathbf{v} = (u, v)$ and a pressure p such that

$$-\eta\nabla^2\mathbf{v} + \nabla p = \mathbf{f}, \tag{7}$$

$$-\nabla.\mathbf{v} = 0, \tag{8}$$

with $\mathbf{v} = \mathbf{0}$ on the boundary of D.

To set up the variational formulation of this problem we need to introduce some function spaces. The velocity is chosen to belong to the space $X = H_0^1(D) \times H_0^1(D)$ and the pressure to the space $M = L_0^2(D)$ where

$$L_0^2(D) = \left\{ \phi \in L^2(D) : \int\!\!\int_D \phi = 0 \right\}.$$

The variational formulation is then given by the following: find $(\mathbf{v}, p) \in X \times M$ such that

$$\eta(\nabla\mathbf{v}, \nabla\mathbf{w}) - (p, \nabla.\mathbf{w}) = (\mathbf{f}, \mathbf{w}) \quad \forall\, \mathbf{w} \in X, \tag{9}$$

$$-(q, \nabla.\mathbf{v}) = 0 \quad \forall\, q \in M. \tag{10}$$

We define two bilinear forms a and b over $X \times X$ and $X \times M$, respectively, by

$$a(\mathbf{v}, \mathbf{w}) = \eta(\nabla\mathbf{v}, \nabla\mathbf{w}), \tag{11}$$

$$b(\mathbf{v}, q) = -(q, \nabla.\mathbf{v}). \tag{12}$$

The Stokes problem can then be written as follows: find $(\mathbf{v}, p) \in X \times M$ such that

$$a(\mathbf{v}, \mathbf{w}) + b(\mathbf{w}, p) = (\mathbf{f}, \mathbf{w}) \quad \forall\, \mathbf{w} \in X, \tag{13}$$

$$b(\mathbf{v}, q) = 0 \quad \forall\, q \in M. \tag{14}$$

In order to set up the corresponding variational formulation it is necessary to choose suitable polynomial subspaces X_N and M_N of X and M, respectively. Let N be an integer which will be used as a discretization parameter for the pseudospectral approximation. Let $P_N(D)$ denote the space of all polynomials of degree less than or equal to N in each spatial direction. The velocity approximation space is chosen to be the subspace of X comprising polynomials of degree at most N in each coordinate direction which vanish on the boundary of D, i.e.,

$$X_N = X \cap P_N(D)^2.$$

If the discrete pressure space is chosen to comprise polynomials of the same degree as for the discrete velocity space then spurious modes in the pressure representation might arise. If there are spurious modes in the pressure representation then the problem does not possess a unique solution [5]. For example, if Chebyshev or Legendre polynomials are used as the basis functions, then for the discrete version of (13) and (14) it can be shown that there are eight spurious modes. In practice these modes pollute the pressure approximation and need to be removed in order to recover an accurate pressure field. Phillips and Roberts [18] filter out the spurious pressure modes using a singular value decomposition of the pressure coefficient matrix. In the case of domain decomposition, however, it is not immediately self-evident how to identify and then remove the spurious modes. The spurious modes have been characterised for a rectangular domain decomposition problem [21]. However, in the general case it is preferable to define a discrete problem which is free from spurious modes in the pressure representation.

One means of circumventing this difficulty is to deplete the pressure space in an appropriate way to remove the spurious modes. Bernardi and Maday [3] propose a method based on a staggered mesh, similar to that used in some finite difference methods, in which the dependent variables are defined on a different collocation grid. Although this removes the spurious modes it does not result in optimal error estimates for either the velocity or the pressure. This method is also cumbersome to use since it is necessary to interpolate the variables between the different grids.

Another method which is free from spurious pressure modes and which does not require an inordinate amount of interpolation is the $P_N - P_{N-2}$ method of Maday, Patera, and Rønquist [17]. In this method the pressure approximation is of degree $N-2$ compared with a velocity approximation of degree N. The unknowns associated with the pressure approximation are located at the interior Gauss–Lobatto points.

Let a_N and b_N be discrete bilinear forms which are, for the moment, unprescribed analogues of the continuous ones defined by (11) and (12), respectively. The discrete variational problem is as follows: find $(\mathbf{v}_N, p_N) \in X_N \times M_N$ such that

$$a_N(\mathbf{v}_N, \mathbf{w}_N) + b_N(\mathbf{w}_N, p_N) = (\mathbf{f}, \mathbf{w}_N)_N \quad \forall\, \mathbf{w}_N \in X_N, \tag{15}$$

$$b_N(\mathbf{v}_N, q_N) = 0 \quad \forall\, q_N \in M_N. \tag{16}$$

Suppose that the discrete bilinear forms a_N and b_N satisfy the following properties:

$$|a_N(\mathbf{u}_N, \mathbf{v}_N)| \le \gamma \parallel \mathbf{u}_N \parallel_{H^1(D)} \parallel \mathbf{v}_N \parallel_{H^1(D)}, \tag{17}$$

$$|b_N(\mathbf{u}_N, q_N)| \le \delta \parallel \mathbf{u}_N \parallel_{H^1(D)} \parallel q_N \parallel_{L^2(D)}, \tag{18}$$

$$a_N(\mathbf{u}_N, \mathbf{u}_N) \ge \alpha N^{-s} \parallel \mathbf{u}_N \parallel^2_{H^1(D)}, \tag{19}$$

$$\inf_{q_N \in M_N} \sup_{\mathbf{v}_N \in X_N} \frac{b_N(\mathbf{v}_N, q_N)}{\parallel \mathbf{v}_N \parallel_{H^1(D)} \parallel q_N \parallel_{L^2(D)}} \ge \beta N^{-t} \tag{20}$$

for all $\mathbf{u}_N, \mathbf{v}_N \in X_N$, and $q_N \in M_N$ and where α, β, γ, and δ are constants independent of N and s and t are nonnegative real numbers. Conditions (17) and (18) state that the bilinear forms a_N and b_N are continuous over $X_N \times X_N$ while (19) is a statement of the ellipticity of a_N over $X_N \times X_N$. Condition (20) is known as the inf-sup or compatibility condition, and if it is satisfied then the discretization spaces X_N and M_N are said to be compatible. If these four conditions are satisfied then the problem (15)–(16) is well posed and yields a unique solution $(\mathbf{v}_N, p_N) \in X_N \times M_N$.

Suppose that the solution $(\mathbf{v}, p)$ belongs to $H^{\sigma}(D)^2 \times H^{\sigma-1}(D)$ for a real number $\sigma \geq 1$ and that $\mathbf{f}$ belongs to $H^{\rho}(D)^2$ for a real number $\rho > 1$; then the following error estimates hold for the velocity and pressure approximations:

$$\| \mathbf{v} - \mathbf{v}_N \|_{H^1(D)} \leq cN^s[N^{1-\sigma}(\| \mathbf{u} \|_{H^{\sigma}(D)^2} + \| p \|_{H^{\sigma-1}(D)}) + N^{r-\rho} \| \mathbf{f} \|_{H^{\rho}(D)}],$$

$$\| p - p_N \|_{L^2(D)} \leq cN^{s+t}[N^{1-\sigma}(\| \mathbf{u} \|_{H^{\sigma}(D)^2} + \| p \|_{H^{\sigma-1}(D)}) + N^{r-\rho} \| \mathbf{f} \|_{H^{\rho}(D)}].$$

The parameter r arises in the estimate of the interpolation error with respect to the given set of collocation points. This error is determined by measuring how closely the inner product $(\mathbf{f}, \mathbf{w}_N)$ is approximated by the discrete inner product $(\mathbf{f}, \mathbf{w}_N)_N$.

In the case of the fully staggered grid approach of Bernardi and Maday [3] we have $r = 1$, $s = 1$, and $t = \frac{5}{2}$. For the method based on the use of a pressure approximation of the same degree as the velocity but with the spurious modes removed [4] we have $r = 1$, $s = 0$, and $t = 1$. Finally, for the method based on a pressure approximation of degree $N - 2$ where N is the corresponding degree of the velocity approximation [17] we have $r = 0$, $s = 0$, and $t = \frac{1}{2}$. This method thus yields an optimal error estimate for the velocity and the best estimate for the pressure amongst the three methods. Therefore, this method has the advantages of being free from spurious pressure modes and yielding the best error estimates for the model problem (7)–(8). The lack of optimality in the pressure approximation is a direct consequence of having a nonzero value of t in the inf-sup condition. This property not only affects the accuracy of the pressure but also results in a bad condition number for the Uzawa operator used to solve the discrete set of equations. This operator results from a decoupling of the pressure and velocity computations. Therefore, from a computational as well as a theoretical point of view the lower the value of t the better. These considerations have motivated our use of the so-called $P_N - P_{N-2}$ method of Maday, Patera, and Rønquist [17] in our work.

We now define the discrete bilinear forms for the $P_N - P_{N-2}$ method. Let ξ_i, $0 \leq i \leq N$ be the set of Gauss–Lobatto–Legendre points in the interval $[-1, 1]$. There exists a unique set of positive real numbers w_i, $0 \leq i \leq N$, such that the integration rule

$$\int_{-1}^{1} \psi(x)\, dx = \sum_{i=0}^{N} w_i \psi(x_i)$$

is exact for all $\psi \in P_{2N-1}[-1, 1]$. The discrete bilinear forms a_N and b_N and the discrete inner product for this formulation are defined by

$$a_N(\mathbf{u}, \mathbf{v}) = \sum_{GL} \nabla\mathbf{u}\, \nabla\mathbf{v} \quad \forall\, \mathbf{u}, \mathbf{v} \in (C^1(D))^2,$$

$$b_N(\mathbf{u}, p) = \sum_{GL} \nabla.\mathbf{u}\, p \quad \forall\, \mathbf{u} \in (C^1(D))^2,\ p \in C^0(D),$$

$$(\mathbf{f}, \mathbf{v})_N = \sum_{GL} \mathbf{f}\, \mathbf{v},$$

where

$$\sum_{GL} \psi = \sum_{i=0}^{N} \sum_{j=0}^{N} w_i w_j\ \psi(\xi_i, \xi_j).$$

4. Time-splitting schemes. Time-splitting schemes have the advantage of enabling the different operators in a system of partial differential equations to be treated by appropriate methods of solution. In the present context time-splitting methods are used as a means of determining the solution of the corresponding steady problem. In this respect they may be viewed as iterative techniques. However, for the dynamic loading problem these schemes will be used in true time-dependent mode to track the transient solution of the journal bearing problem, i.e., to determine the locus of the journal in time. In general, nonlinear operators such as the convection operator are treated explicitly while linear operators such as the diffusion, gradient, and divergence operators are treated implicitly.

4.1. Backward Euler. In this scheme the diffusion and pressure computations are coupled. After the spatial discretization has been performed, the velocity and pressure are decoupled by means of an Uzawa method which essentially involves a block Gaussian elimination method to obtain a pressure system. The solution of this system requires the use of a nested iterative solution technique. An efficient preconditioner for this system can only be constructed easily in the case of constant viscosity since otherwise the coefficient matrix of the inner system needs to be updated for each iteration.

STAGE A

$$\frac{\rho}{\Delta t}(\mathbf{v}^* - \mathbf{v}^n) = -\frac{\rho}{\Delta t}\mathbf{v}^n.\nabla\mathbf{v}^n, \tag{21}$$

STAGE B

$$\frac{\rho}{\Delta t}(\mathbf{v}^{n+1} - \mathbf{v}^*) = -\nabla p^{n+1} + 2\nabla.\eta^n\mathbf{d}^{n+1}, \tag{22}$$

$$\nabla.\mathbf{v}^{n+1} = 0. \tag{23}$$

The solution of Stage B requires the inversion of an unsteady Stokes operator at each time step subject to the given velocity boundary conditions on the bearing and the journal. This is the computationally intensive part of the algorithm which we will be investigating in this paper. There is also an alternative backward Euler time splitting method in which the diffusion term is evaluated in Stage A at the starred time level. This scheme has the advantage of decoupling the diffusion and pressure terms completely, so that the corresponding discrete problem does not require a nested iterative solver. However, this scheme suffers from a splitting error of $O(\Delta t)$ which does not vanish as the steady state is reached. This drawback is overcome by use of the Crank–Nicolson scheme, which we describe next [16].

4.2. Crank–Nicolson scheme. In this scheme the treatment of the diffusion and the pressure terms are decoupled. This means that when the scheme is discretized in space there is no requirement for a nested solver. This method is therefore more suitable for variable viscosity problems. The splitting error in this Crank–Nicolson scheme is $O(\Delta t^2)$, and at steady state this error vanishes in contrast to the corresponding backward Euler scheme. Although the discrete pressure operator derived in this case may appear to be nested, the spectral element mass matrix is, in fact, diagonal, and therefore the calculation of its inverse requires no iteration.

5. The spectral element method. In this section we describe the spectral element method applied to Stage B of the backward Euler scheme. The spectral element method is based on a weak formulation of the problem. The variational formulation of (22) and (23) is therefore as follows: find $(\mathbf{v}, p) \in X \times M$ such that

$$\int\!\!\int_\Omega (\eta\nabla\mathbf{v}\nabla\mathbf{w})d\mathbf{x} + \frac{\rho}{\Delta t}\int\!\!\int_\Omega(\mathbf{wv})d\mathbf{x} - \int\!\!\int_\Omega(p\nabla.\mathbf{w})d\mathbf{x} = \frac{\rho}{\Delta t}\int\!\!\int_\Omega(\mathbf{wv}^*)d\mathbf{x}\ \forall\ \mathbf{v} \in X, \tag{24}$$

$$\iint_{\Omega} (\nabla.\mathbf{v}q)d\mathbf{x} = 0 \; \forall \; q \in L^2(\Omega), \tag{25}$$

where we have dropped the time level $n+1$ from the superscript on $\mathbf{v}$ and p.

The spectral element method is a technique for solving problems defined in complex geometries [11], [12]. The idea is to divide the flow into several spectral elements, Ω_k, $1 \leq k \leq K$, such that $\cup_{k=1}^{K} \bar{\Omega}_k = \bar{\Omega}$ and $\Omega_k \cap \Omega_l = \emptyset$ for all $k \neq l$. We also assume that the decomposition is geometrically conforming in the sense that the intersection of two adjacent elements is either a common vertex or an entire edge. Each of the spectral elements is mapped onto the parent element $D = [-1,1] \times [-1,1]$ using the transfinite mapping technique of Gordon and Hall [10].

Since each element Ω_k is mapped onto the parent element $[-1,1] \times [-1,1]$ we may associate with each point $(\xi, \zeta) \in D$ a point $(x(\xi,\zeta), y(\xi,\zeta)) \in \Omega_k$. Under this mapping we have, for example,

$$a^k(\mathbf{v},\mathbf{w}) = \iint_{\Omega_k} \eta \; \nabla\mathbf{v} \; \nabla\mathbf{w} d\mathbf{x} \tag{26}$$

$$= \iint_D \frac{\eta}{J^k} \left\{ G_1^k \frac{\partial \mathbf{v}}{\partial \xi}\frac{\partial \mathbf{w}}{\partial \xi} + G_2^k \frac{\partial \mathbf{v}}{\partial \zeta}\frac{\partial \mathbf{w}}{\partial \zeta} + G_3^k \left(\frac{\partial \mathbf{v}}{\partial \xi}\frac{\partial \mathbf{w}}{\partial \zeta} + \frac{\partial \mathbf{v}}{\partial \zeta}\frac{\partial \mathbf{w}}{\partial \xi} \right) \right\} d\xi \, d\zeta, \tag{27}$$

where the Jacobian J^k of the mapping is defined by

$$J^k = \frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \zeta} - \frac{\partial x}{\partial \zeta}\frac{\partial y}{\partial \xi}, \tag{28}$$

$$G_1 = \left(\frac{\partial x}{\partial \zeta}\right)^2 + \left(\frac{\partial y}{\partial \zeta}\right)^2, \tag{29}$$

$$G_2 = \left(\frac{\partial x}{\partial \xi}\right)^2 + \left(\frac{\partial y}{\partial \xi}\right)^2, \tag{30}$$

$$G_3 = -\left(\frac{\partial x}{\partial \xi}\frac{\partial x}{\partial \zeta} + \frac{\partial y}{\partial \xi}\frac{\partial y}{\partial \zeta} \right). \tag{31}$$

We approximate the primitive variables using Legendre–Lagrangian interpolants of degree N in both spatial directions. Let $P_{N,K}$ denote the space of polynomials of degree N or less, defined over the K elements. We choose the velocity field in $P_{N,K}$ and construct a Gauss–Lobatto–Legendre grid in each of the elements Ω_k, $1 \leq k \leq K$. The velocity representation is then given by

$$\mathbf{v}_N^k(\xi,\zeta) = \sum_{i=0}^{N}\sum_{j=0}^{N} \mathbf{v}_{i,j}^k h_i(\xi) h_j(\zeta), \tag{32}$$

where the Lagrangian interpolants $h_i(\xi)$, $0 \leq i \leq N$, are defined on the parent interval with $\xi \in [-1,1]$ by the relationship

$$h_i(\xi) = -\frac{(1-\xi^2)L_N'(\xi)}{N(N+1)L_N(\xi_i)(\xi - \xi_i)}, \tag{33}$$

where the points ξ_i are the collocation points on the Gauss–Lobatto–Legendre grid.

As for the finite element method, the velocity and pressure approximations must satisfy the Babuška–Brezzi condition to avoid the presence of spurious modes. In the framework of

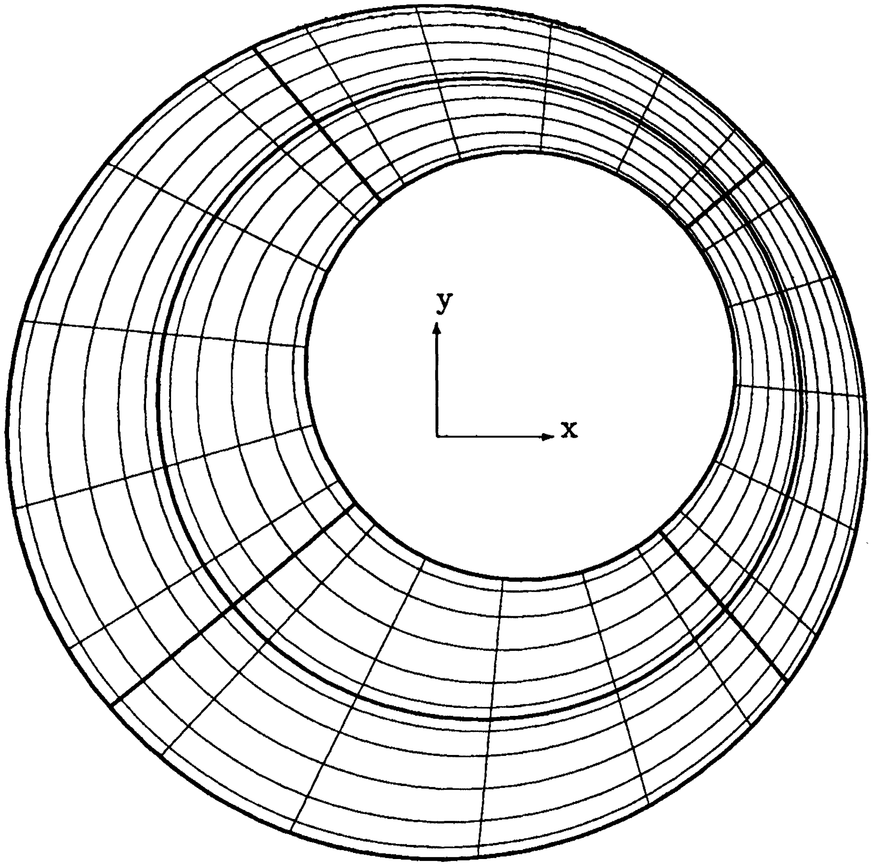


FIG. 2. *The discretization of the physical fluid domain for four elements in the azimuthal direction* ($N_a = 4$) *and two elements in the radial direction* ($N_r = 2$). *Each element is split into seven nodes in each direction* ($N = 7$).

the spectral element method Maday and Patera [15] have shown that a suitable choice for the pressure approximation space is

$$M_N = L^2(\Omega) \cap P_{N-2,K} \ . \tag{34}$$

Thus the pressure approximation is given by

$$p_N^k(\xi, \zeta) = \sum_{i=1}^{N-1} \sum_{j=1}^{N-1} p_{i,j}^k \hat{h}_i(\xi) \hat{h}_j(\zeta), \tag{35}$$

where $\hat{h}_i(\xi)$, $1 \leq i \leq N - 1$, are the Lagrangian interpolants based on the interior Gauss–Lobatto–Legendre points.

The spectral element discretization of the geometry is shown in Figure 2 together with the Gauss–Lobatto collocation grid. A typical collocation grid on the parent element is shown in Figure 3. The number of spectral elements in the radial and azimuthal directions is denoted by E_r and E_a, respectively. A full description and algorithm of the grid generation is given in Appendix A.

The integrals over each element Ω_k in (26) are approximated by a Gauss–Legendre quadrature rule.

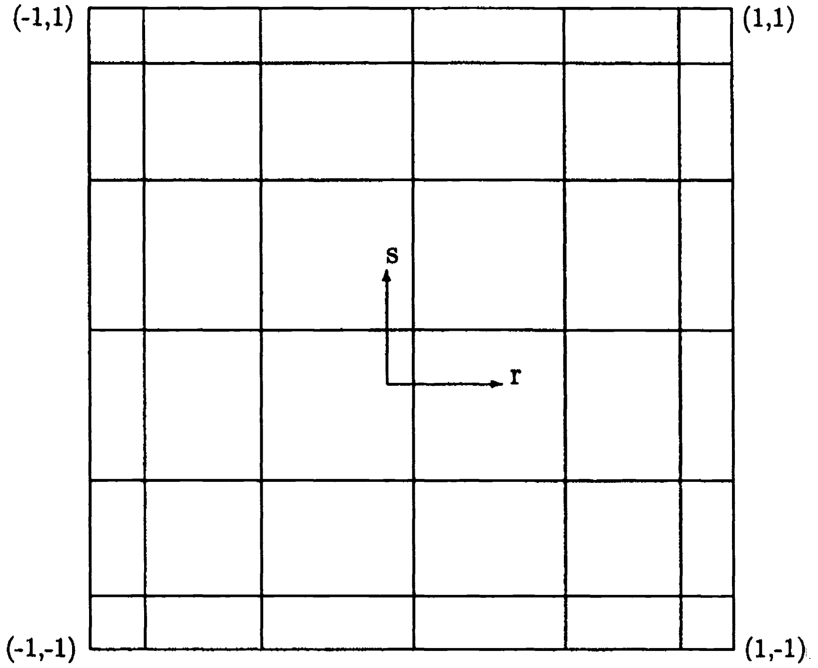


FIG. 3. *The discretization of the parent element into the N^2 nodes ($N = 7$) according to a Legendre–Gauss–Lobatto distribution.*

The velocity and pressure expansions are now inserted into the discrete form of (24)–(25), and the discrete equations are generated by choosing appropriate test functions $\mathbf{w} \in P_{N,K} \cap H_0^1(\Omega)$ which are unity at a single point (ξ_k, ζ_l) and zero at all other Gauss–Lobatto–Legendre points and test functions $q \in P_{N-2,K}$ which are unity at a single point (ξ_k, ζ_l) and zero at all other interior Gauss–Lobatto–Legendre points. Once the discrete bilinear forms have been computed for all elements, the contributions from neighbouring elements are summed along element interfaces. This procedure takes into account the situations in which the test function extends into more than one element. In this way we obtain the system of algebraic equations

$$A\mathbf{v} + \sigma B\mathbf{v} - D^T\mathbf{p} = \mathbf{f}, \tag{36}$$

$$D\mathbf{v} = \mathbf{g}, \tag{37}$$

where D is the discrete divergence operator, its transpose is the discrete gradient operator, B is the mass matrix, and $\sigma = \rho/\Delta t$. The vectors of unknowns now refer to their values at the discretization points (the Gauss–Lobatto nodes). Note that in order to evaluate the discrete gradient and divergence operators we need to map the pressure p_N and the test function q_N within each spectral element from the interior Gauss–Lobatto–Legendre nodes onto the Gauss–Lobatto–Legendre nodes on the boundary of the element.

Block Gaussian elimination yields a symmetric semipositive definite system for the pressure unknowns

$$S\mathbf{p} = \mathbf{c}, \tag{38}$$

where $S = D(A + \sigma B)^{-1}D^T$ and $\mathbf{c} = -D(A + \sigma B)^{-1}\mathbf{f} + \mathbf{g}$. The matrix S is made positive definite by imposing a zero volume condition on the pressure field. The velocity is computed using

$$(A+\sigma B)\mathbf{v} = D^T p+\mathbf{f}. \tag{39}$$

Due to the properties of S and its considerable size an iterative solver to solve (38) is sought. For the model problem on the square grid it can be shown that the maximum eigenvalue of S is fixed and is of order unity while the minimum eigenvalue scales like β_N^{-2}. Hence there is a necessity from the solution point of view to have a good inf-sup condition. Using the preconditioned conjugate gradient (PCG) method to solve this system is well documented [14], and, using a suitable preconditioner, the PCG method is found to be very efficient. For a wide range of problems [14]–[17] this preconditioner works very well within the spectral element context, and we would not advocate the use of anything else. However, for the solution of realistic journal bearing problems the system of equations is extremely ill conditioned due to the nature of the geometry. To illustrate this more clearly we will discuss our problem applied to the constant viscosity steady Stokes solver. The iterative scheme is nested since each matrix–vector multiplication $S\mathbf{p}$ requires the solution of the system

$$(A+\sigma B)\mathbf{r} = \mathbf{s}. \tag{40}$$

This inner system is also solved by the conjugate gradient method since it is a symmetric positive definite system.

6. Preconditioners. The convergence rate of iterative methods for solving a system of linear algebraic equations depends on the spectrum of the coefficient matrix. The idea underlying the preconditioning philosophy is to transform the original system into one that is equivalent in the sense that it has the same solution but that has a more favourable spectrum. We are interested in finding a matrix P, known as the preconditioner, which approximates the coefficient matrix S in some sense so that the transformed system

$$P^{-1}S\mathbf{p} = P^{-1}\mathbf{b} \tag{41}$$

has the same solution as (38) has but the eigenvalues of $P^{-1}S$ are more clustered than those of S. The trade-off between the cost of constructing and applying the preconditioner and the expected gain in convergence speed of the iterative method must be borne in mind. The transformation of the linear system (41) is not what is actually used in the computations. Instead, the preconditioner is decomposed in the form $P = QQ^T$ and the transformed system written as

$$Q^{-1}SQ^{-T}(Q^T\mathbf{p}) = Q^{-1}\mathbf{b}. \tag{42}$$

The condition number, κ, of the preconditioned system is defined by

$$\kappa = \frac{\lambda_{\max}}{\lambda_{\min}} \tag{43}$$

where $\lambda_{\max}$ and $\lambda_{\min}$ are the largest and smallest eigenvalues of the preconditioned matrix L where

$$L = Q^{-1}SQ^{-T}, \quad P = QQ^T. \tag{44}$$

A given iterative method is then preconditioned following this procedure:

1. The right-hand side vector is transformed according to

$$\tilde{\mathbf{b}} = Q^{-1}\mathbf{b}.$$

2. Apply the iterative method to the system

$$L\tilde{\mathbf{p}} = \tilde{\mathbf{b}}.$$

3. Compute

$$\mathbf{p} = Q^{-T}\tilde{\mathbf{b}}.$$

The preconditioner is chosen to be an approximation to S, in some sense, which is easier and cheaper to invert than S. Ideally the preconditioner should have similar properties to the original matrix and also be sparse so that it is efficient to construct and to store.

The decomposition of P given in (44) is not needed in practice. The steps of the conjugate gradient method can be rewritten so that the preconditioner is applied in its entirety [9].

6.1. Constant viscosity problem.

6.1.1. Steady Stokes solver. As mentioned previously, direct methods for solving (38) are undesirable due to the considerable time required in preprocessing S. Furthermore, due to the bad conditioning of S for small α we find that Choleski decomposition gives inaccurate results. In the present context accuracy is measured by comparing our results with those given in [18] and also knowledge that steady Stokes flow is symmetric with respect to the line joining the centres of the journal and the bearing. We find that preconditioning is absolutely necessary for the efficient and accurate solution of this problem.

Here we are interested in finding effective preconditioners for the matrix $S = \eta DA^{-1}D^T$. This matrix has rank equal to the number of pressure degrees of freedom, i.e., $M = K(N-1)^2$, and will be completely full due to the presence of A^{-1}. Maday and Patera [15] demonstrate that for some test problems the matrix S is extremely well conditioned. They use a heuristic argument to explain this phenomenon. They argue that the spectrum of S behaves like the variational equivalent of the identity operator since it involves the product of gradients "divided" by the Laplacian. This leads them to suggest that the pressure mass matrix would make an ideal preconditioner. For a wide range of problems [14]–[17] this preconditioner works very well within the spectral element context, and we would not advocate the use of anything else. However, we present results which show that the condition number of S is geometry dependent; i.e., for the distorted spectral elements in the physical journal bearing problem the spectrum of S is not so universally well behaved as Maday and Patera [15] perhaps suggest.

The way the spectral element grid is constructed means that as far as the geometrical parameters of the problem are concerned the entries of the pressure matrix S are dependent on the eccentricity ϵ but not on the orientation of the journal. We denote this dependence by S_ϵ. The following preconditioners were considered for the constant viscosity steady Stokes problem:

1. $P = B$ where B is the pressure mass matrix. This matrix is not sparse at the elemental level.
2. $P = \hat{B}$ where $\hat{B}$ is the diagonal of B.
3. $P = S_{\epsilon^*}$ where S_{ϵ^*} is the matrix S_ϵ evaluated at $\epsilon = \epsilon^*$.

The geometry of a typical car engine journal bearing is such that the physical aspect ratios, which are proportional to $\alpha \equiv c/2\pi R_J$, are very small where $c \equiv R_B - R_J$ is the average gap of the fluid region. Typical values are much less than $\frac{1}{100}$. Therefore, not even a significant redefinition of the spectral element discretization would overcome this problem.

We show that the eigenvalue spectrum is very much dependent on the physical aspect ratio. In Figures 4 and 5 we choose $\epsilon = 0.5$ and keep (N, E_r, E_a) fixed at $(7, 2, 2)$, which gives $M = 196$, and show the eigenvalue spectra of the preconditioned system when the

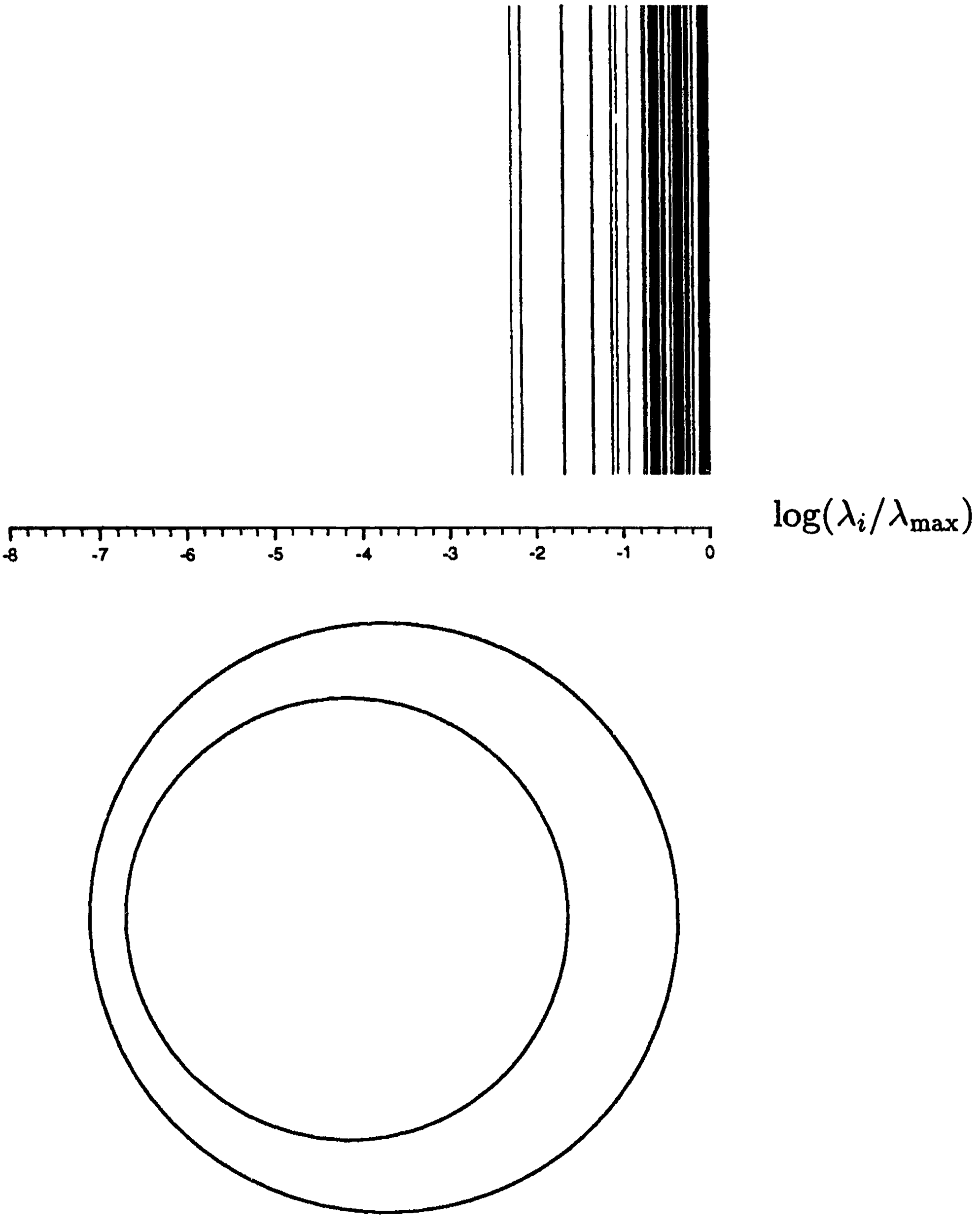


FIG. 4. *Eigenvalue spectrum of a preconditioned system with* $P = B$ *with* $(N, E_r, E_a) = (7, 2, 2)$ *and an illustration of the model for the case of* $\alpha = 5.3 \times 10^{-2}$, $\epsilon = 0.5$. *The result is a condition number* $\kappa = 188.3$ *and the number of PCG iterations* = 28.

preconditioner is the pressure mass matrix, i.e., $P = B$. The outer radius in both figures is the same ($R_B = 0.03129$) and the inner radius is varied so as to vary α.

In Figure 4, $\alpha = 0.053$ whilst in Figure 5, $\alpha = 2.04 \times 10^{-4}$. One can see from Figure 5 that the value of α is so small that the two cylinders are indistinguishable from each other. The result is that the condition number increases as α decreases with a cluster of very small eigenvalues appearing in the latter figure. In terms of conjugate gradient iterations the model with $\alpha = 0.053$ takes 28 iterations to converge whilst with $\alpha = 2.04 \times 10^{-4}$, 40 iterations

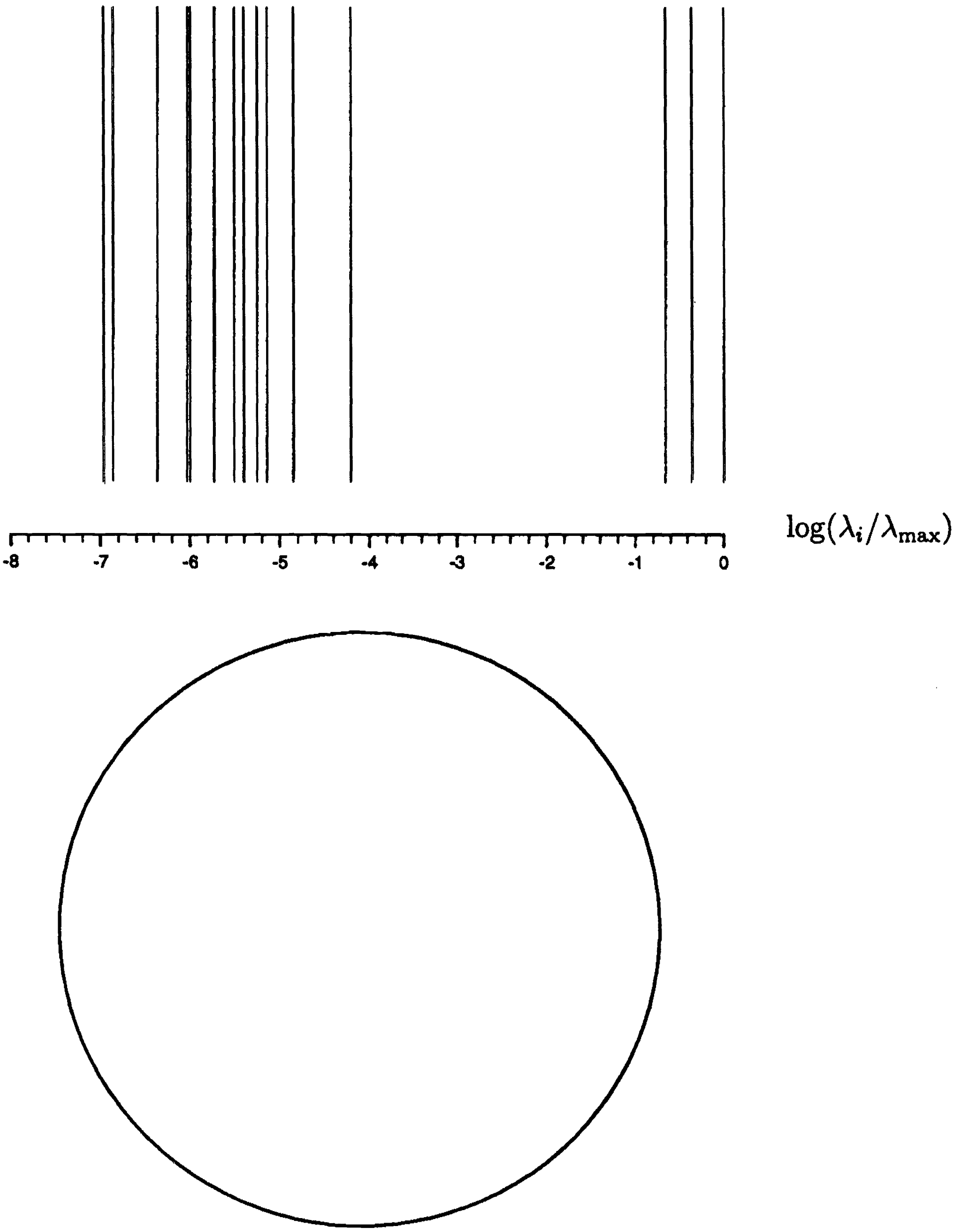


FIG. 5. *Eigenvalue spectrum of a preconditioned system with* $P = B$ *and* $(N, E_r, E_a) = (7, 2, 2)$ *together with an illustration of the model for the case of* $\alpha = 2.04 \times 10^{-4}$, $\epsilon = 0.5$. *The result is a condition number* $\kappa = 9.35 \times 10^6$ *and the number of PCG iterations* = 40.

are required. The position was much worse for the preconditioner $P = \hat{B}$ for which the PCG method failed to converge within the theoretical maximum number of M iterations for $\alpha = 2.04 \times 10^{-4}$, indicating that round-off errors dominate.

Changing the number of elements or the number of nodes whilst keeping the elemental aspect ratio fixed does not alter the condition number of the preconditioned system with $P = B$. This is illustrated in Figures 6 and 7 where the eigenvalue spectra are given for the same model as illustrated in Figure 5 except that $(N, E_r, E_a) = (7, 4, 4)$ and $(12, 2, 2)$,

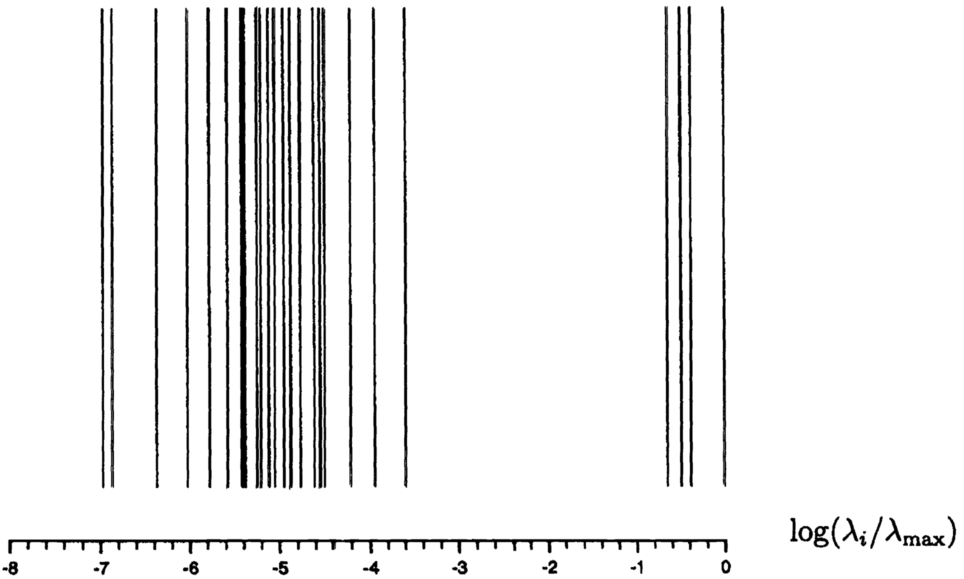


FIG. 6. *Eigenvalue spectrum of a preconditioned system with $P = B$ and $(N, E_r, E_a) = (7, 4, 4)$ for the case of $\alpha = 2.04 \times 10^{-4}$, $\epsilon = 0.5$. The result is a condition number $\kappa = 9.36 \times 10^6$.*

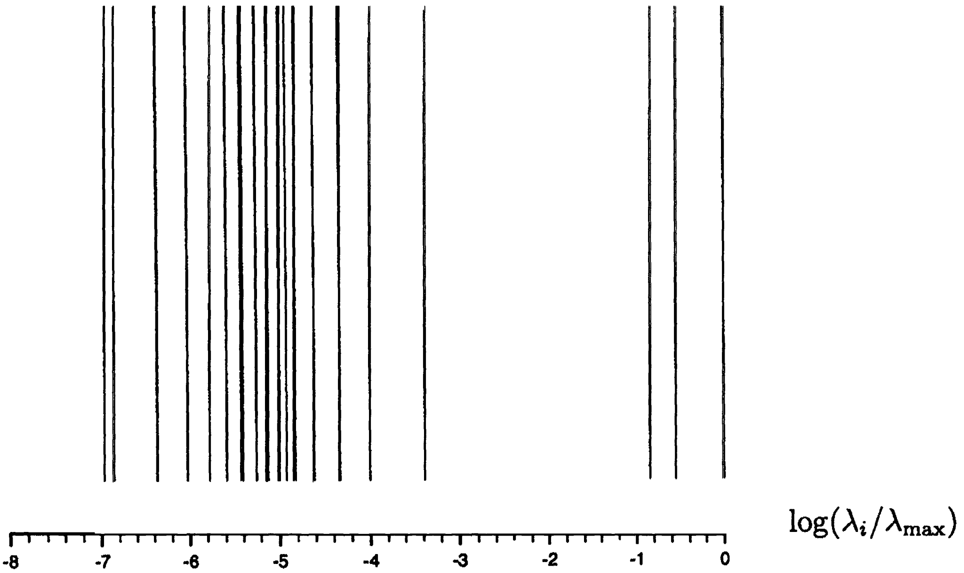


FIG. 7. *Eigenvalue spectrum of a preconditioned system with $P = B$ and $(N, E_r, E_a) = (12, 2, 2)$ for the case of $\alpha = 2.04 \times 10^{-4}$, $\epsilon = 0.5$. The result is a condition number $\kappa = 9.36 \times 10^6$.*

respectively, as opposed to (7,2,2). The resulting condition number for both these examples is $\kappa = 9.36 \times 10^6$.

Thus we have shown conclusively that the condition number of the preconditioned system is dependent on the preconditioner and the physical aspect ratio, α, but not on the elemental aspect ratio. Figures 8 and 9 give the eigenvalue spectra for two different elemental aspect ratios corresponding to (7, 2, 4) and (7, 4, 2), respectively, whilst α is kept fixed at 2.04×10^{-4}. The condition number remains unaltered with a value of $\kappa = 9.36 \times 10^6$.

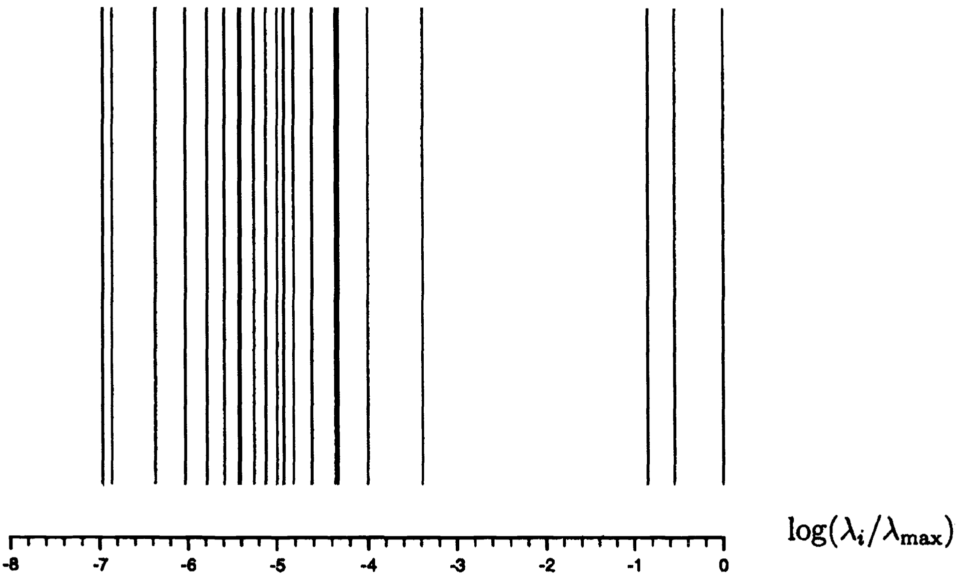


FIG. 8. *Eigenvalue spectrum of preconditioned system with* $P = B$ *and* $(N, E_r, E_a) = (7, 2, 4)$ *for the case of* $\alpha = 2.04 \times 10^{-4}$, $\epsilon = 0.5$. *The result is a condition number* $\kappa = 9.36 \times 10^6$.

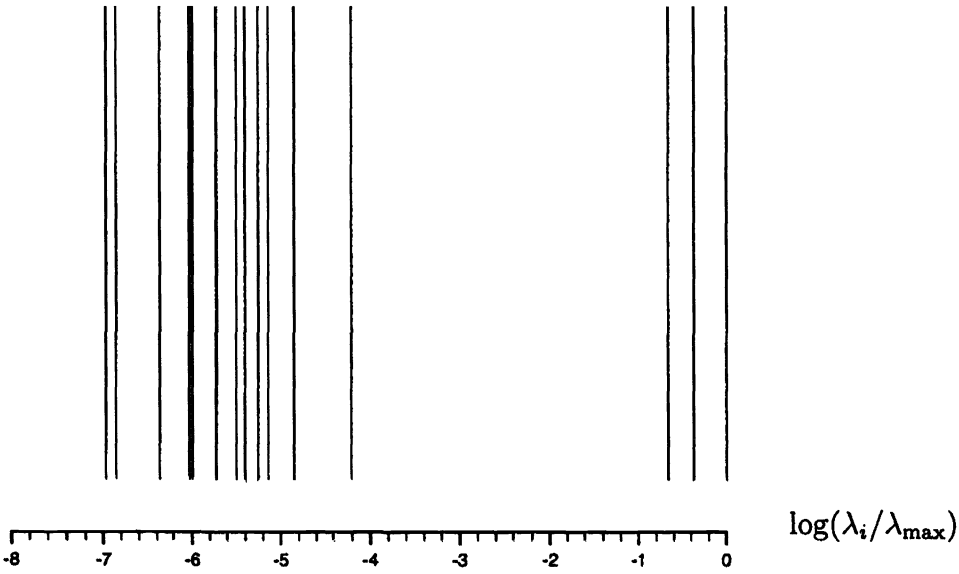


FIG. 9. *Eigenvalue spectrum of a preconditioned system with* $P = B$ *and* $(N, E_r, E_a) = (7, 4, 2)$ *for the case of* $\alpha = 2.04 \times 10^{-4}$, $\epsilon = 0.5$. *The result is a condition number* $\kappa = 9.36 \times 10^6$.

The number of eigenvalues in the cluster of small eigenvalues is the same as the number of pressure nodes N_a in the azimuthal direction, where $N_a = N \times E_a$. This is also the case for the eigenvalue spectrum of the unpreconditioned matrix S. For the unpreconditioned system the eigenvectors corresponding to these small eigenvalues display a pattern. Irrespective of the eccentricity of the system, these "zero eigenvectors" can be associated with a set of nodal points P_k which lie on the same radial line as the kth nodal point. As the average gap c decreases, the values of the "zero eigenvalues" become smaller and the corresponding "zero

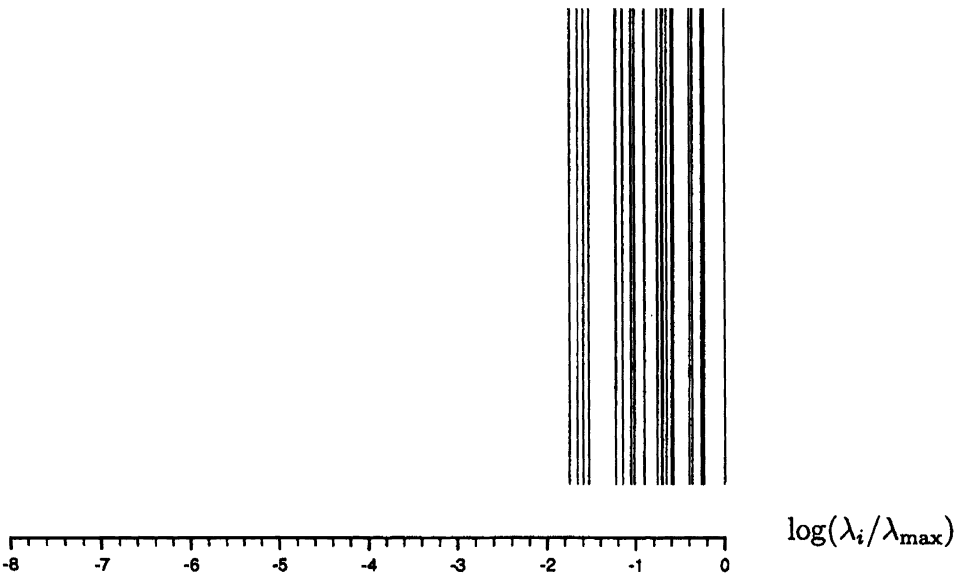


FIG. 10. *Eigenvalue spectrum of a preconditioned system with* $P = S_{0.0}$ *and* $(N, E_r, E_a) = (7, 2, 2)$, $\alpha = 2.04 \times 10^{-4}$, *and* $\epsilon = 0.5$. *The result is a condition number* $\kappa = 55.8$ *and the number of PCG iterations* $= 27$.

eigenvectors" tend toward the vectors $\mathbf{e}_k$, $k = 1, \ldots, N_a$, whose entries are given by

$$\mathbf{e}_k(i) = 0 \text{ if } i \notin P_k,$$
$$\mathbf{e}_k(i) = 1 \text{ if } i \in P_k.$$

This behaviour is independent of the eccentricity of the system. This leads us to choose as preconditioner the pressure matrix $S_{0.0}$ that is generated from the concentric model ($\epsilon = 0.0$). If the gap c is small then the structure of the "zero eigenvalues" for the two models (S, $S_{0.0}$) are similar. The resulting eigenvalue spectrum with $P = S_{0.0}$ is illustrated in Figure 10, and this can be compared directly with the spectrum given in Figure 5 where $P = B$.

The following tables compare the condition number, the number of PCG iterations, and CPU time per PCG iteration for different preconditioners for S in the case when $\epsilon = 0.5$. It is clear that the most successful preconditioner in terms of number of PCG iterations required for convergence is $S_{0.0}$. There are also advantages with respect to accuracy for using this preconditioner since its condition number is considerably lower than that of its competitors. Even with respect to CPU time, preconditioners based on nondiagonal matrices are only slightly more expensive to use than those based on diagonal matrices. The reason for this somewhat surprising statement is that the CPU times are dominated by the inner iteration process. The number of inner iterations is independent of the choice of preconditioner in the outer iteration. The only significant difference in times occurs for larger values of M when the cost of using a dense preconditioner is about 25% more than the cost of using a diagonal preconditioner. However, given the considerably fewer number of iterations required for convergence in the former case, the difference in CPU times is not enough to justify a diagonal preconditioner. The preconditioner based on $S_{0.0}$ needs to be calculated only once and can be applied to systems with different eccentricities. The inversion of the pressure mass matrix also requires a nontrivial amount of work. In addition it needs to be recomputed for different eccentricities. The other two preconditioners (I and $\hat{B}$), although trivial to calculate and invert, are very poor preconditioners in terms of the number of PCG iterations required

TABLE 1a

The number of PCG iterations of the preconditioned system with $\epsilon = 0.5$ for different preconditioners P together with, in brackets, the corresponding condition number. The results are for different numerical parameters based upon the p-method analysis. $\alpha = 2.04 \times 10^{-4}$.

$(N, E_r, E_a), M$	$P = I$	$P = \hat{B}$	$P = B$	$P = S_{0.0}$
(7,2,2), 196	284 (4.30×10^8)	164 (1.79×10^8)	40 (9.35×10^6)	29 (55.84)
(10,2,2), 400	1162 (8.50×10^8)	395 (3.31×10^8)	63 (9.35×10^6)	34 (55.84)
(13,2,2), 676	>2000 (1.39×10^9)	727 (5.24×10^8)	95 (9.35×10^6)	37 (55.84)

TABLE 1b

The CPU times, in seconds, of one PCG iteration of the preconditioned system with $\epsilon = 0.5$ for different preconditioners P. The results are for different numerical parameters based upon the p-method analysis. $\alpha = 2.04 \times 10^{-4}$.

$(N, E_r, E_a), M$	$P = I$	$P = \hat{B}$	$P = B$	$P = S_{0.0}$
(7,2,2), 196	0.8	0.8	0.8	0.8
(10,2,2), 400	6.2	6.2	7.3	7.3
(13,2,2), 676	22.0	22.0	26.0	26.0

for convergence. In practice we are not limited to using the preconditioner based on the concentric coefficient matrix $S_{0.0}$. It is possible to construct and store a set of preconditioners S_{ϵ_i}, $1 \le i \le I$, for different eccentricities ϵ_i and to use the preconditioner S_{ϵ_j} for which ϵ_j is closest to ϵ.

Note that the condition number of the preconditioned system is independent of N for the two cases $P = B$ and $P = S_{0.0}$. In this sense we say that both preconditioners are optimal and that we can expect the number of PCG iterations to be proportional to M. However, it can be seen from Table 1 that the number of iterations using both these preconditioners increases less than in proportion with N. This effect is more pronounced in the case $P = S_{0.0}$, where the number of iterations required for $N = 10$ and $N = 13$ is almost the same.

6.1.2. Unsteady Stokes solver. For the unsteady Stokes problem with a constant viscosity η the resulting pressure matrix S_t is given by

$$S_t = D(\eta A + \sigma B)^{-1} D^T, \tag{45}$$

where $\sigma = \rho/\Delta t$. The "natural well conditioning" that exists in the steady pressure matrix S does not exist in S_t. The matrix S_t is again symmetric positive definite, and, as in the steady case, the solution to the linear system

$$S_t \mathbf{p} = \mathbf{r} \tag{46}$$

can be found using a nested PCG iteration.

A preconditioner considered in [7] is a linear combination of the inverses of the identity and the Laplacian operators. This choice is motivated in [14] by considering the two limits of S_t as $\Delta t \to 0$ and $\Delta t \to \infty$; that is,

$$S_t \to \eta^{-1} S \text{ as } \Delta t \to \infty, \tag{47}$$

$$S_t \to \sigma^{-1} E \text{ as } \Delta t \to 0, \tag{48}$$

where $E = DB^{-1}D^T$ is the discrete Poisson operator. The resulting preconditioner is written as

$$P^{-1} = \eta B^{-1} + \sigma E^{-1}. \tag{49}$$

TABLE 2

The number of PCG iterations and the condition number of the preconditioned system for the two different preconditioners corresponding to equations (50) *and* (51), *respectively. The parameters of the problem are* $(N, E_r, E_a) = (7, 2, 2)$*; that is,* $M = 196$ *for the case of* $\epsilon = 0.5$, $\alpha = 2.04 \times 10^{-4}$, $\eta = 5$ *(mPa s),* $\rho = 820 kg/\mathrm{m}^3$.

	$\epsilon^* = 0.0$	$\epsilon^* = 0.25$	$\epsilon^* = 0.5$
$\Delta t = 0$	100(2.28 × 10^7)	81(1.47×10^7)	40(9.35 × 10^6)
$\Delta t = 10^{-3}$	79(540.4)	55(324.2)	36(180.7)
$\Delta t = 10^{-4}$	48(57.3)	36(34.4)	25(19.2)
$\Delta t = 10^{-5}$	35(9.09)	28(6.04)	18(4.47)
$\Delta t = 10^{-7}$	25(6.29)	20(4.52)	11(2.65)

	$\epsilon^* = 0.0$	$\epsilon^* = 0.25$	$\epsilon^* = 0.5$
$\Delta t = 0$	29(55.8)	17(9.94)	1(1.00)
$\Delta t = 10^{-3}$	28(25.6)	17(5.65)	4(1.03)
$\Delta t = 10^{-4}$	27(18.1)	17(4.68)	7(1.11)
$\Delta t = 10^{-5}$	23(6.75)	16(2.97)	7(1.23)
$\Delta t = 10^{-7}$	22(4.26)	16(2.38)	7(1.13)

This preconditioner clearly has the advantage that in adaptive time-stepping schemes the recalculation of P^{-1} for different time steps is trivial. As in the steady solver this preconditioner would need to be recalculated if the eccentricity of the model changed. We generalize preconditioners of the form (49) as follows:

$$P^{-1} = \eta B_{\epsilon^*}^{-1} + \sigma E_{\epsilon^*}^{-1}, \tag{50}$$

where the subscript ϵ^* means that the relevant matrices are evaluated at that particular eccentricity ratio. The preconditioner defined in (50) is identical to that defined in (49) when $\epsilon^* = \epsilon$. We compare the use of (50) with

$$P^{-1} = \eta S_{\epsilon^*}^{-1} + \sigma E_{\epsilon^*}^{-1}, \tag{51}$$

which is the unsteady problem preconditioner analogous to $P = S_{\epsilon^*}$ for the steady problem.

Table 2 displays the number of PCG iterations required for the preconditioners given by (50) and (51). Also shown in this table, in brackets, is the condition number of the preconditioned system.

It is clear that the preconditioner involving the full steady pressure matrix S_{ϵ^*}, i.e., (51), is superior to that using the pressure mass matrix B_{ϵ^*}, i.e., (50). The extent to which (51) is a better preconditioner than (50) depends upon the time step. For a small Δt the two preconditioners are at their most comparable with the system matrix being closer to σE; when $\Delta t = 10^{-7}$, the preconditioner in (50) for $\epsilon^* = 0.5, 0.25$ is still better than that in (51) at $\epsilon^* = 0.25, 0.0$, respectively. However, for larger time steps, when the steady Stokes operator becomes more influential, the performance of the preconditioner in (51) becomes increasingly better when compared with (50). For example, for time steps of greater than approximately 10^{-4} the performance of (51) for the concentric geometry ($\epsilon^* = 0.0$) outperforms (50) for $\epsilon^* = \epsilon = 0.5$.

For solving (46) at a number of different eccentricities we would choose a limited number of eccentricities and for each one calculate the preconditioner from (51). The number of preconditioners that need to be calculated and for which eccentricities is dependent on the number of solutions to the pressure system (46) that is required and the range of eccentricity ratios. In a typical dynamic model as many as $O(10^5)$ solutions are performed with eccentricities

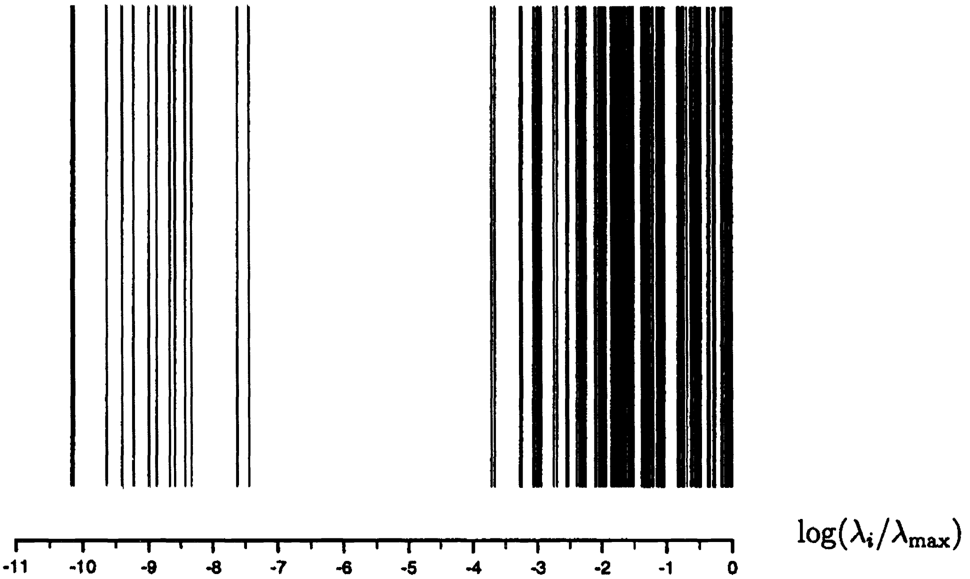


FIG. 11. *Eigenvalue spectrum of DD^T and $(N, E_r, E_a) = (7, 2, 2)$ for the case of $\alpha = 2.04 \times 10^{-4}$, $\epsilon = 0.5$. The result is a condition number $\kappa = 1.49 \times 10^{10}$.*

varying almost throughout the whole domain, i.e., $\epsilon \in [0, 1)$. For such a model the calculation of 10 preconditioners at intervals of equal eccentricity is found to be both effective and cheap in comparison with the total run time. Clearly, the number of iterations listed in Table 2 and in the rest of the paper is only for the comparison of different methods and preconditioners; the actual number of iterations can be reduced dramatically, for example, by a choice of the initial guess based upon knowledge of the solution at other eccentricities.

6.2. Variable viscosity problem. For the unsteady non-Newtonian problem difficulties arise with finding preconditioners for the resulting Schur complement matrix S^η which is now defined as

$$S^\eta = D(A_\eta + \sigma B_v)^{-1} D^T, \tag{52}$$

where A_η represents the discrete weak formulation of the diffusion operator $\eta\Delta$. For the constant viscosity case this operator was represented simply by ηA. For the corresponding variable viscosity problem this is no longer the case. If there are only slight variations in the viscosity then the preconditioner associated with the constant viscosity problem may be reasonable with effective viscosity $\hat{\eta}$ which is an average, in some sense, of the variable viscosity throughout Ω. However, in a typical high-speed journal bearing problem the viscosity can vary by as many as two orders of magnitude due to, primarily, the variation in pressure. Difficulties arise in the numerical solution process because the viscosity is now embedded in the operator A_η, and this, in turn, is nested between the multiplication of the gradient and divergence operators. For problems in which there is a large variation in viscosity there is no obvious reason why the *average viscosity* preconditioner should work well.

The poor conditioning of S, S^t, and S^η derives from the poor conditioning of DD^T, whose eigenvalue spectrum is shown in Figure 11. The Crank–Nicolson scheme described in §4.2 provides a mechanism for removing the variable viscosity from the nested solution of systems with coefficient matrices of the form (52). The resulting linear algebraic problem involves

TABLE 3

Information on the viscosity of the steady-state fluid for geometry of $\epsilon = 0.5$ and $\alpha = 2.04 \times 10^{-4}$ for angular velocity $\omega = 250$, numerical parameters $(8, 2, 2)$, and viscosity parameters given in Appendix B (η_A & η_B). All units are in (mPa s).

	η_A	η_B
$\langle\hat{\eta}\rangle$	4.9996	4.9993
$\eta_{\max}$	8.3061	9.5120
$\eta_{\min}$	4.7350	2.9996

two solutions of a system of equations of the form

$$H\mathbf{r} = \mathbf{s}, \tag{53}$$

where H is the discrete Helmholtz operator defined by

$$H = A_\eta + \sigma B_v \tag{54}$$

and one solution of a system of the form

$$S^o\mathbf{r} = \mathbf{t}, \tag{55}$$

where

$$S^o = DB_v^{-1}D^T \tag{56}$$

is a Schur complement matrix and B_v is the velocity mass matrix.

Equation (55) can be solved using a PCG method. This is not a nested iteration since the matrix B_v is diagonal and hence the calculation of its inverse is trivial. The preconditioner used is simply $S^o_{\epsilon*}$ since S^o is independent of both the variable viscosity and the time step. For the PCG method applied to (53) we use as preconditioner $\hat{H}$ the diagonal of H which can be readily calculated and inverted.

We compare the Crank–Nicolson scheme with preconditioners defined above with the scheme described in the previous section, in which the Schur complement matrix (52) is preconditioned by

$$P = \hat{\eta}S_{\epsilon*}^{-1} + \sigma E_{\epsilon*}^{-1}, \tag{57}$$

where $\hat{\eta}$ is the average viscosity of the fluid.

At a first glance it would seem that the Crank–Nicolson method involves more work, requiring three matrix solutions as compared with the one solution in the case of solving (52). However, the inversion of the system with coefficient matrix given by (52) requires a nested iteration, the inner one of which may require a large number of iterations on systems of the form (53) to converge.

We refer to the two different approaches as method-S (Schur) and method-CN (Crank–Nicolson), respectively. In comparing the performance of method-S with method-CN we require different viscosity fields. We use the standard geometry used previously defined by $\epsilon = 0.5$ and $\alpha = 2.04 \times 10^{-4}$. In addition, we use the viscosity generated for the steady flow between the cylinders with the inner cylinder rotating at an angular velocity of 250 rad/s about its own axis.

Table 3 illustrates the range of the viscosities for the two viscosity fields we have chosen, η_A and η_B. The parameters which define these viscosity functions are given in Appendix B. The viscosity parameters are chosen so that the average viscosities are similar in order to make as fair a comparison as possible.

Clearly the viscosity relating to η_B has much more variation than that relating to η_A. For this reason, a preconditioner based upon a constant viscosity is expected to be more successful

TABLE 4

The number of PCG iterations, with the condition number in parentheses, of the preconditioned system using method-S for inverting (52). *The data for the model corresponds to that in Table* 3, *and we have taken* $\epsilon^* = \epsilon$. *In addition to the two variable viscosity fields the result for the constant viscosity case of* $\eta_O = 5$ *mPa s is listed. In all cases* $\rho = 820kg/\mathrm{m}^3$.

	η_A	η_B	η_O
$\Delta t = 10^{-1}$	11(1.58)	20(2.92)	3(1.02)
$\Delta t = 10^{-3}$	11(1.57)	20(2.92)	4(1.03)
$\Delta t = 10^{-5}$	11(1.53)	17(2.84)	7(1.23)
$\Delta t = 10^{-7}$	9(1.30)	14(2.47)	7(1.13)

for the η_A model than for the η_B model, and this is shown to be the case. The performance of method-S, in terms of number of outer PCG iterations, is illustrated in Table 4 together with the associated condition numbers. In this table we have taken $\epsilon^* = \epsilon = 0.5$.

Although neither of the two viscosities has a range covering the two orders of magnitude reported earlier in this paper, the variation is enough to illustrate the problem that can arise with variable viscosity, with a large variation increasing the number of iterations for the method-S. A variation of viscosity covering two orders of magnitude can be obtained if the eccentricity of the journal is very large or if the journal has a large squeeze (translational) velocity.

The conclusion here is that variation in viscosity causes an increase in the number of PCG iterations required for convergence, and this is most evident when σ is small and hence the contribution of A_η in (54) is large. For the variable viscosity problem it is therefore not surprising that as Δt decreases the number of iterations decreases. The opposite behaviour is observed for the constant viscosity case since for the extreme values of Δt with $\epsilon = \epsilon^*$ the preconditioner tends toward the system matrix (47), (48) and the range $\Delta t \in (10^{-5}, 10^{-7})$ corresponds roughly to a peak in the number of such iterations.

In this paper we have not mentioned the number of inner iterations required for these nested schemes. The inner problem in the nested solver associated with method-S is the same as the first two solvers required for the method-CN, that is, (53). For both these cases we compare the performance of two preconditioners, namely the diagonal of the matrix, that is, $P = \hat{H}$ and $P = \hat{\eta} A_{\epsilon^*}$, with $\epsilon^* = \epsilon = 0.5$. Clearly we could have chosen a less optimal ϵ^* for the latter preconditioner but Tables 5a and 5b should only be taken as a guide since more work needs to be done on the preconditioning of this system. The number of velocity degrees of freedom is denoted by M_v. There are surprising results in these tables, notably in Table 5a where the number of iterations does not follow the decrease in condition number. As Δt decreases, σ increases, and hence we would expect H to tend toward σB, thus making $\hat{H}$ a more powerful preconditioner. This is supported by the condition number of the system. The reason the number of iterations does not decrease with the condition number is due to the eigenvalue distribution of A. For a very small aspect ratio the eigenvalues of the preconditioned system of A with $P = \hat{A}$ are clustered into well-defined distinct groups. The band width of each cluster is dependent on α, and each of the $(N+1) \times E_r - 1$ clusters contains $(N+1) \times E_a$ eigenvalues. To illustrate this we present the eigenvalue spectrum of this system in Figure 12 in which, to the naked eye, the system has only $(N + 1) \times E_r - 1$ distinct eigenvalues. In comparison, we have in Figure 13 the eigenvalue spectrum of the preconditioned system of H with $P = \hat{H}$ now with $\Delta t = 10^{-5}$. The PCG theory [6] states that, in exact arithmetic, the number of PCG iterations required to converge to the correct solution is equal to at most the number of distinct eigenvalues. This is almost the case when $\Delta t = 0$ and explains why so few iterations are required for a system which has a large condition number.

For the $\Delta t = 10^{-5}$ case each local cluster of eigenvalues has spread out and hence, despite the lower condition number, the number of distinct eigenvalues has increased dramatically, thus increasing the maximum number of iterations required for convergence.

TABLE 5a

The number of PCG iterations and the condition number of the preconditioned system for the preconditioner $\hat{H}$ applied to the problem of (53). *The parameters of the problem are* $(N, E_r, E_a) = (7, 2, 2)$; *that is,* $M_v = 240$, $\alpha = 2.04 \times 10^{-4}$, $\rho = 820 kg/\mathrm{m}^3$.

	η_O	η_A	η_B
$\Delta t = 0$	30(243.6)	69(259.14)	68(255.73)
$\Delta t = 10^{-1}$	33(243.5)	70(259.11)	69(255.71)
$\Delta t = 10^{-3}$	56(242.1)	75(256.94)	74(254.31)
$\Delta t = 10^{-5}$	75(150.8)	105(156.33)	113(165.58)
$\Delta t = 10^{-7}$	22(7.5)	25(7.41)	26(8.72)

TABLE 5b

The number of PCG iterations and the condition number of the preconditioned system for the preconditioner $\hat{\eta}A$ *applied to the problem of* (53). *The parameters of the problem are* $(N, E_r, E_a) = (7, 2, 2)$; *that is,* $M_v = 240$, $\alpha = 2.04 \times 10^{-4}$, $\rho = 820 kg/\mathrm{m}^3$.

	η_O	η_A	η_B
$\Delta t = 0$	1(1.00)	10(1.68)	16(3.06)
$\Delta t = 10^{-1}$	3(1.01)	10(1.68)	16(3.06)
$\Delta t = 10^{-3}$	5(1.06)	10(1.68)	16(3.07)
$\Delta t = 10^{-5}$	18(6.98)	21(7.37)	26(11.47)
$\Delta t = 10^{-7}$	100(556.7)	127(573.3)	130(827.4)

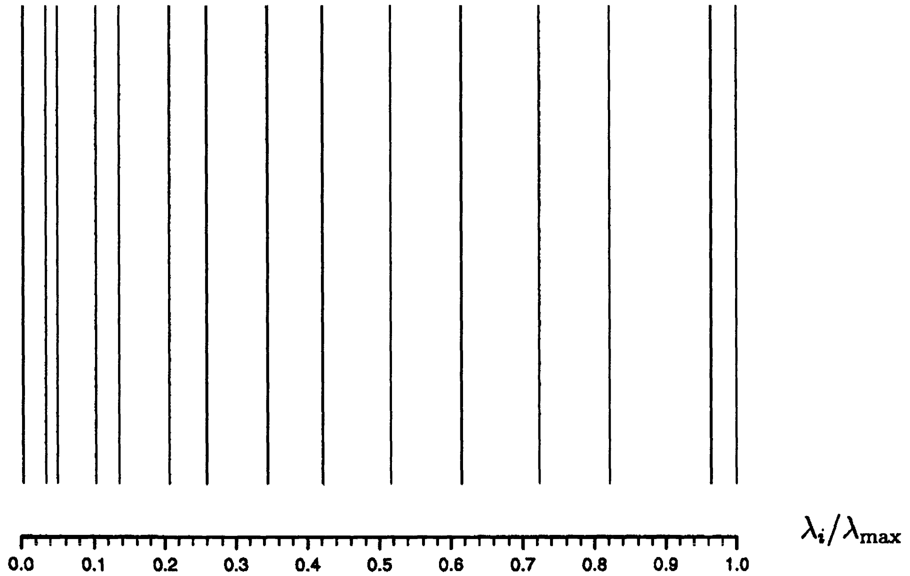


FIG. 12. *Eigenvalue spectrum of a preconditioned system of H with* $P = \hat{H}$, *constant viscosity* $\eta = 5$ *mPa s*, $\Delta t = 0$, $\alpha = 2.04 \times 10^{-4}$, $(N, E_r, E_a) = (7, 2, 2)$, *and* $\epsilon = 0.5$. *The result is a condition number* $\kappa = 243.59$ *and the number of PCG iterations* $= 30$.

The same argument can be applied to the case of variable viscosity, although the band width of each local eigenvalue cluster is greater than in the constant viscosity case. For $\Delta t = 10^{-7}$ the Helmholtz operator is sufficiently near to the diagonal mass matrix for the diagonal preconditioner to be effective.

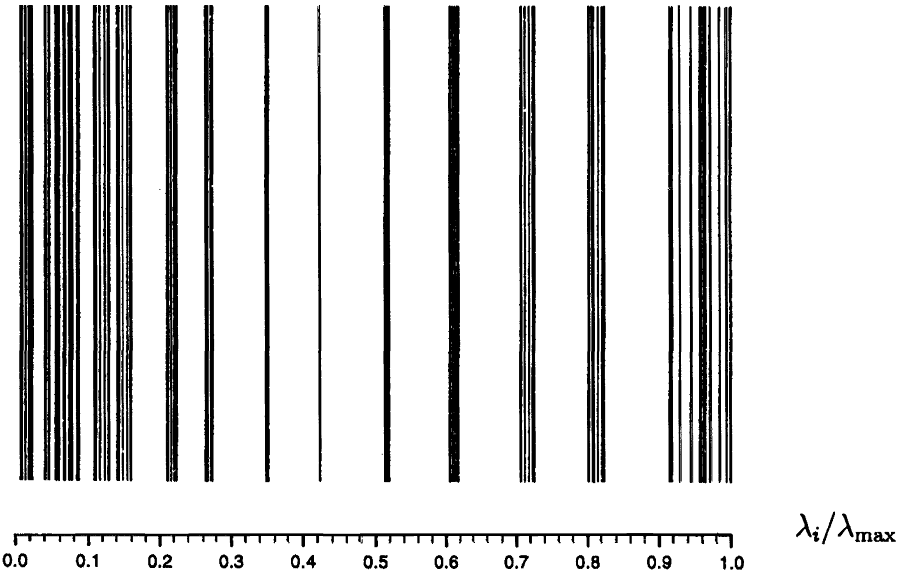


FIG. 13. *Eigenvalue spectrum of a preconditioned system of H with $P = \hat{H}$, constant viscosity $\eta = 5$ mPa s, $\rho = 820 kg/m^3$, $\Delta t = 10^{-5}$, $\alpha = 2.04 \times 10^{-4}$, $(N, E_r, E_a) = (7, 2, 2)$, and $\epsilon = 0.5$. The result is a condition number $\kappa = 150.76$ and the number PCG iterations $= 75$.*

TABLE 6

The number of PCG iterations and the condition number of the preconditioned system for the preconditioner $P = S^o_{\epsilon^*}$ applied to the problem of (56) for various ϵ^* and $\epsilon = 0.5$. The parameters of the problem are $(N, E_r, E_a) = (7, 2, 2)$; that is, $M = 196$, $\alpha = 2.04 \times 10^{-4}$.

$\epsilon^*= 0.25$	$\epsilon^*= 0.4$	$\epsilon^*= 0.45$	$\epsilon^*= 0.5$	$\epsilon^*= 0.55$	$\epsilon^*= 0.6$
21(2.26)	15(1.44)	12(1.21)	3(1.00)	12(1.24)	16(1.57)

The results of Table 5b are much as expected with the number of iterations and the condition number increasing with decreasing Δt as the Helmholtz operator departs from the Laplacian operator. We have shown that the choice of preconditioner is dependent on the time step. For practical applications we must also consider the efficiency of using $P = \hat{\eta} A_{\epsilon^*}$ for $\epsilon^* \neq \epsilon$.

Table 6 shows the performance of various preconditioners for the third stage of the Crank–Nicolson scheme, i.e., equation (55). This equation is independent of the variable viscosity and of the time step and hence all there is to vary is ϵ^*.

To make a fair comparison between method-CN and method-S we should first of all compare the results of Table 4 with those of Table 6 under the column $\epsilon^* = 0.5$, that is, $\epsilon^* = \epsilon$. The rest of the entries in Table 6 are a guide to how many preconditioners need be calculated to be within a given tolerance in the number of iterations.

Purely in terms of the number of iterations required it is clear that using method-CN is more effective than using method-S. This is because for method-S, if we use $\epsilon^* = \epsilon$ then the preconditioner is, in fact, S^o. The reason why as many as three iterations are required to solve for (56) in this case is that S^o is so badly conditioned that the resulting Choleski decomposition is inexact.

At each outer iteration of method-S nested solutions need to be performed which are of the same form as the first two time stages of the Crank–Nicolson scheme. Therefore, if the number of outer iterations in solving the system with the coefficient matrix given by (52) exceeds two then the Crank–Nicolson scheme is the quickest in this context. Further, the number of iterations required to invert (56) is less than the number of outer iterations to invert

(52). We make no comment here on the accuracy of the two approaches, which also needs to be taken into account in making a final comparison between the two methods.

7. Conclusions. In this paper we have developed efficient iterative methods for solving the two-dimensional flow of a non-Newtonian lubricant between eccentrically rotating cylinders. The constitutive model employed is a generalized Newtonian fluid with a viscosity which is shear thinning and pressure thickening. A spectral element method is used for the spatial discretization of the governing equations and first- and second-order time-splitting schemes for the time integration.

For the constant viscosity problem we have considered a first-order backward Euler scheme for the time integration. This couples the velocity and pressure computations. They are decoupled algebraically using an Uzawa-type method. The iterative solution of the resulting pressure system requires a nested iteration. Both inner and outer systems are solved using the PCG method.

When the spectral elements are not highly distorted we would advocate the use of the pressure mass matrix as preconditioner. However, we have shown that even for the steady constant viscosity problem the conditioning of the linear systems generated by the spectral element discretization is poor in the extreme case when the radii of the journal and the bearing are close. This is the situation which occurs in practice. A robust preconditioner for this problem is the pressure matrix for the corresponding concentric cylinder problem. The advantage of this preconditioner over the one based on the pressure mass matrix is that it is only necessary to calculate it once since it can be applied to systems with different eccentricities.

For the unsteady Stokes problem a preconditioner is constructed so that the calculation of its inverse is trivial if the time step is changed. This approach commends itself for adaptive time-stepping schemes. Again preconditioners based on the pressure matrix evaluated at a specific eccentricity are used to precondition systems defined for a range of eccentricities.

Finally, for the variable viscosity problem the velocity and pressure are completely decoupled using a Crank–Nicolson scheme in order to extricate the viscosity function from the inner iterative process. It is important to do this since the viscosity may vary greatly and thus an updated preconditioner may be required after a certain number of time steps. This is, of course, an expensive process. The Crank–Nicolson time-splitting scheme alleviates the need for this extra computation. Preconditioners are given for each of the velocity and pressure systems based on ideas explored for the constant viscosity problem.

Future work will concentrate on the extension of these ideas to the dynamically loaded journal bearing. In this problem we will seek to determine the minimum oil film thickness in the journal bearing due to a prescribed applied load on the journal.

Appendix A: Grid generation. This appendix explains how the grid in Figure 2 is generated. The number of elements in the azimuthal and radial direction (E_a,E_r) are known and this determines the elemental boundaries.

All the radial lines are congruent on the centre of the journal, and this includes the interelemental radial lines. We insist upon E_r being even, and this allows us to set the line joining the centre of the journal C_J and the centre of the bearing C_B as two of the interelemental radial lines. The rest of such lines are drawn so that they are equidistributed in an angle about the centre of the journal. The other radial lines are drawn so that they are distributed in between the interelemental radial lines in such a way that they are distributed in an angle about the centre of the journal in a Gauss–Lobatto distribution.

The distribution of the azimuthal lines is similar in idea to the distribution of radial lines. Every azimuthal line constitutes a circle, and these circles are defined by their centres and corresponding radii. All the centres lie on the line joining C_J and C_B and lie in between them. The interelemental circles have their centres equidistributed between C_J and C_B; the other circles have their centres distributed between the centres of the interelemental circles in a Gauss–Lobatto distribution. The radius of the interelemental circles are equidistributed between the radius of the journal and the radius of the bearing. The rest of the circles have their radii distributed between the radii of the interelemental circles in a Gauss–Lobatto distribution.

The discretization nodes constitute the intersection of all these lines with the journal and bearing's surface also constituting such lines.

The above description can be written in algorithmic form. Let N, E_a, and E_r be as defined in the main part of the paper. We label the azimuthal lines from 0 to N_a, where $N_a = N \times E_a$, and this includes the journal and bearing's boundary. The radial lines are labelled from 0 to N_r, where $N_t = (N \times E_a)$, with the zeroth and N_rth lines representing the same line. If we define the zeroth line as the line joining the centres, extended to the region of smallest gap, then the following uniquely determines the point distribution.

Let the Gauss–Lobatto point distribution be denoted by the vector

$$\mathbf{g}(i) \in [-1, 1], \quad i \in [0, N],$$

the centres and radii of the circles denoted by the respective vectors

$$\mathbf{c}(i) \in [C_J, C_B], \quad i \in [0, N_a],$$

$$\mathbf{r}(i) \in [R_J, R_B], \quad i \in [0, N_a],$$

and the angle distribution about C_j of the radial lines denoted by the vector

$$\mathbf{a}(i) \in [0, 2\pi], \quad i \in [0, N_r].$$

Then the following algorithm holds

```
begin
for $e = 1, E_a$
    step=$2 \times (E_a - 1)$
    for $i = 0, N$
        $v = \mathbf{g}(i)$+step
        $j = i + (e - 1) \times N$
        $r = (v + 1)/(2 \times E_a)$
        $\mathbf{c}(j) = r \times e$
        $\mathbf{r}(j) = R_J + r \times (R_B - R_J)$
    continue
continue

for $e = 1, E_r$
    step=$2 \times (E_r - 1)$
    for $i = 0, N$
        $v = \mathbf{g}(i)$+step
        $j = i + (e - 1) \times N$
        $r = (v + 1)/(2 \times E_r)$
        $\mathbf{a}(j) = r \times 2\pi$
    continue
continue
end
```

where e is the eccentricity.

Appendix B: Viscosity parameters. The parameter values taken for the two variable viscosity fields η_A and η_B when applied to (4) are shown in Table B1.

TABLE B1

	η_A	η_B	Units
η_0	9.352×10^{-4}	9.352×10^{-4}	Pa s
η_∞	4.500×10^{-4}	4.500×10^{-4}	Pa s
α	1.119×10^{-8}	2.000×10^{-7}	Pa^{-1}
$\bar{\alpha}$	2.390×10^{-8}	5.000×10^{-7}	Pa^{-1}
m	-21.33	-21.33	
E	7.902	7.902	
F	2.380	2.301	

Acknowledgements. We wish to thank A. R. Davies and G. W. Roberts of the University of Wales for many fruitful and stimulating discussions.

REFERENCES

[1] H. A. BARNES, J. F. HUTTON, AND K. WALTERS, *An Introduction to Rheology*, Elsevier, Amsterdam, 1989.

[2] T. W. BATES, B. WILLIAMSON, J. A. SPEAROT, AND C. K. MURPHY, *A Correlation Between Engine Oil Rheology and Oil Film Thickness in Engine Journal Bearing*, Paper 860376, Society of Automotive Engineers, Warrendale, PA, 1986.

[3] C. BERNARDI AND Y. MADAY, *A collocation method over staggered grids for the Stokes problem*, Internat. J. Numer. Methods Fluids, 8 (1988), pp. 537–557.

[4] C. BERNARDI, C. CANUTO, AND Y. MADAY, *Generalized inf-sup condition for Chebyshev spectral approximation of the Stokes problem*, SIAM J. Numer. Anal., 25 (1988), pp. 1237–1271.

[5] C. BERNARDI, Y. MADAY, AND B. METIVET, *Calcul de la pression dans la résolution spectrale du problème de Stokes*, La Researche Aérospatiale, 1 (1987), pp. 1–21.

[6] G. BIRKHOFF AND R. E. LYNCH, *Numerical Solution of Elliptic Problems*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1984.

[7] J. CAHOUET AND J.-P. CHABARD, *Some fast* 3D *finite element solvers for the generalized Stokes problem*, Internat. J. Numer. Methods Fluids, 8 (1988), pp. 869–895.

[8] M. J. DAVIES AND K. WALTERS, *The behaviour of non-Newtonian lubricants in journal bearings—a theoretical study*, in Rheology of Lubricants, T. C. Davenport, ed., Applied Science Publishers, Barking, UK, 1972, pp. 65–80.

[9] G. GOLUB AND C. VAN LOAN, *Matrix Computations*, The Johns Hopkins Press, Baltimore, MD, 1989.

[10] W. J. GORDON AND C. A. HALL, *Construction of curvilinear coordinate systems and application to mesh generation*, Internat. J. Numer. Methods Engrg., 7 (1973), pp. 461–477.

[11] K. Z. KORCZAK AND A. T. PATERA, *Isoparametric spectral element method for solution of the Navier-Stokes equations in complex geometry*, J. Comput. Phys., 62 (1986), pp. 361–382.

[12] P. F. FISCHER AND A. T. PATERA, *Parallel spectral element solution of the Stokes problem*, J. Comput. Phys., 92 (1991), pp. 380–421.

[13] X. K. LI AND A. R. DAVIES, *Numerical modelling of nonisothermal viscoelastic flow between eccentrically rotating cylinders*, in Proc. IUTAM Symposium on Numerical Simulation of Nonisothermal Viscoelastic Flows, Kluwer, The Netherlands, 1995, pp. 159–177.

[14] Y. MADAY, D. MEIRON, A. T. PATERA, AND E. M. RØNQUIST, *Analysis of iterative methods for the steady and unsteady Stokes problem: Application to spectral element discretizations*, SIAM J. Sci. Comput., 14 (1993), pp. 310–337.

[15] Y. MADAY AND A. T. PATERA, *Spectral element methods for the incompressible Navier-Stokes equations*, in State of the Art Surveys in Computational Mechanics, A. K. Noor and J. T. Oden, eds., American Society of Mechanical Engineers, New York, 1989, pp. 71–143.

[16] Y. MADAY, A. T. PATERA, AND E. M. RØNQUIST, *An operator-integration-factor splitting method for time-dependent problems. Application to incompressible fluid flow*, J. Sci. Comput., 5 (1990), pp. 263–292.

[17] ———, *The* $P_N - P_{N-2}$ *method for the approximation of the Stokes problem*, Numer. Math., (1996), to appear.

[18] T. N. PHILLIPS AND G. W. ROBERTS, *The treatment of spurious pressure modes in spectral incompressible flow calculations*, J. Comput. Phys., 105 (1993), pp. 150–164.

[19] G. W. ROBERTS, A. R. DAVIES, AND T. N. PHILLIPS, *Three-dimensional spectral approximations to Stokes flow between eccentrically rotating cylinders*, Internat. J. Numer. Methods Fluids, 13 (1991), pp. 217–233.

[20] G. W. ROBERTS AND K. WALTERS, *On viscoelastic effects in journal-bearing lubrication*, Rheol. Acta, 31 (1992), pp. 55–62.

[21] G. SAACHI LANDRIANI AND H. VANDEVEN, *A multidomain spectral collocation method for the Stokes problem*, Numer. Math., 58 (1990), pp. 441–464.

SIAM J. SCI. COMPUT.
Vol. 17, No. 6, pp. 1395–1413, November 1996

A HIERARCHICAL DOMAIN DECOMPOSITION PRECONDITIONER FOR *h-p* FINITE ELEMENT APPROXIMATION ON LOCALLY REFINED MESHES*

MARK AINSWORTH†

Abstract. A domain decomposition preconditioner suitable for h-p finite element approximation on locally refined meshes with nonuniform polynomial degree is proposed and analysed. The preconditioner is highly suited for parallel computation. The algorithm generalizes methods proposed by Smith [*Domain Decomposition Algorithms for the Partial Differential Equations of Linear Elasticity*, Ph.D. thesis, Mathematics Department, New York University, New York, 1991] (for the h-version finite element method with piecewise linear basis functions) and by Mandel [*Comput. Methods Appl. Mech. Engrg.*, 80 (1990), pp. 117–128] (for the p-version finite element method). The analysis shows that the condition number of the preconditioned system grows at most as $\min(H/h, 1+\log^2 p)(1+\log^2(Hp/h))$. This result generalizes the known estimates in each of the special cases mentioned above. Numerical examples are given confirming the theoretical analysis.

Key words. h-p version finite element method, preconditioning, domain decomposition, hierarchical basis method

AMS subject classifications. 65N30, 65F10

1. Introduction.

Let Ω be a bounded domain in $\mathbb{R}^2$ with piecewise smooth boundary $\partial\Omega$. Consider the second-order elliptic problem

$$-\sum_{j,k=1}^{2} \frac{\partial}{\partial x_j}\left(a_{jk}\frac{\partial U}{\partial x_k}\right) = f \text{ in } \Omega \tag{1}$$

subject to $U = 0$ on the boundary $\partial\Omega$. The data a_{jk} is assumed to be uniformly positive definite, bounded, and piecewise smooth on Ω. The bilinear form $B : H^1(\Omega) \times H^1(\Omega) \mapsto \mathbb{R}$ is given by

$$B(U,V) = \sum_{j,k=1}^{2} \int_\Omega a_{jk}\frac{\partial U}{\partial x_j}\frac{\partial V}{\partial x_k}\,\mathbf{dx} \tag{2}$$

where $H^1(\Omega)$ is the usual Sobolev space of distributions with square integrable derivatives. The subspace $H_0^1(\Omega) \subset H^1(\Omega)$ is the completion of smooth functions with support in Ω, with respect to the $H^1(\Omega)$-norm. The variational formulation of the problem defined by (1) is

$$\text{Find } U \in H_0^1(\Omega) \text{ such that } B(U,V) = (f,V) \text{ for all } V \in H_0^1(\Omega). \tag{3}$$

1.1. Coarse grid.

The finite element approximation of problem (3) will be developed by first subdividing the domain Ω into an initial "coarse" partitioning $\mathcal{P}_H$ consisting of nonoverlapping quadrilateral elements $\{\Omega_H^K\}$ such that

$$\overline{\Omega} = \bigcup_{\Omega_H^K \in \mathcal{P}_H} \overline{\Omega}_H^K \tag{4}$$

and for $K \neq J$, $\overline{\Omega}_H^K \cap \overline{\Omega}_H^J$ is either empty or an entire side or a common vertex of the domains. Let

$$H_K = \operatorname{diam} \Omega_K, \quad H = \max_K H_K \tag{5}$$

*Received by the editors September 13, 1994; accepted for publication (in revised form) June 10, 1995.

†Mathematics Department, Leicester University, Leicester LE1 7RH, U.K. (ain@mcs.le.ac.uk).

and

$$\rho_K = \sup\{\operatorname{diam} \mathcal{B} : \mathcal{B} \text{ is a ball in } \Omega_K\}. \tag{6}$$

It is supposed that there exist positive constants σ, τ independent of h such that

$$\frac{H}{H_K} \leq \tau, \quad \frac{H_K}{\rho_K} \leq \sigma. \tag{7}$$

The partitioning $\mathcal{P}_H$ is therefore quasiuniform of size $O(H)$. The space X_H consists of piecewise bilinear functions defined on the partitioning $\mathcal{P}_H$. An approximation of problem (1) can then be obtained by solving the discrete problem

$$\text{Find } u_H \in X_H \text{ such that } B(u_H, v_H) = (f, v_H) \text{ for all } v_H \in X_H. \tag{8}$$

If the data a_{jk} is discontinuous then the initial coarse partitioning $\mathcal{P}_H$ is constructed so that the element edges coincide with the interfaces along which the data is discontinuous. Such a choice has a beneficial effect on the accuracy of the approximation u_H and will have a significant impact on the preconditioning algorithm to be developed. Typically, owing to the geometry of the domain (e.g., re-entrant corners) and discontinuities in the data a_{jk}, the true solution U of problem (1) will be nonsmooth in certain regions. It is often necessary to enhance the approximation space X_H in the neighbourhood of those areas where the true solution is less regular.

1.2. Types of refinements. Various strategies have been proposed as to how to improve the accuracy. The simplest techniques are to either subdivide the mesh uniformly throughout the domain (the h-version finite element method) or to enrich the polynomial degree of the functions on each of subdomains (the p-version finite element method [4]). Alternatively, one can use a posteriori error estimates to identify specific regions in which the approximation is poor and perform a selective enrichment by locally subdividing the elements and increasing the polynomial degree nonuniformly. The latter approach is in the spirit of the h-p version of the finite element method [3, 5]. A concrete strategy for producing meshes with selective h and p refinement is given in [12] and a general data structure supporting such refinements is given in [8].

The chief advantage of the uniform refinement strategies is that the data structure is kept simple, meaning that the algorithms can often be coded extremely efficiently. However, the disadvantages are that degrees of freedom are introduced in regions where the solution can be adequately represented using the initial coarse approximation. Conversely, the h-p version can lead to approximations in which the required accuracy is obtained using a minimal number of degrees of freedom. Unfortunately, a price is paid in supporting a much more elaborate data structure both in terms of coding effort and the efficiency of the implementation.

1.3. Locally uniform refinements. One can contemplate retaining some advantages of both approaches by adopting a refinement strategy based on refining the approximation space in a *locally uniform* manner. Such a scheme allows the use of local refinements to resolve local features of the solution and at the same time retain a relatively simple data structure. Specifically, we shall consider classes of mesh obtained by subdividing specific elements in the coarse mesh into a uniform mesh of subelements of size $O(h)$. Global continuity is preserved by constraining the new basis functions obtained by the refinement on the boundaries between the parent elements. Naturally, should a pair of neighbouring coarse elements be refined, then the degrees of freedom on the common interface can be left unconstrained. This type of refinement scheme has been employed, for example, in the work of Ewing et al. [6]. Equally

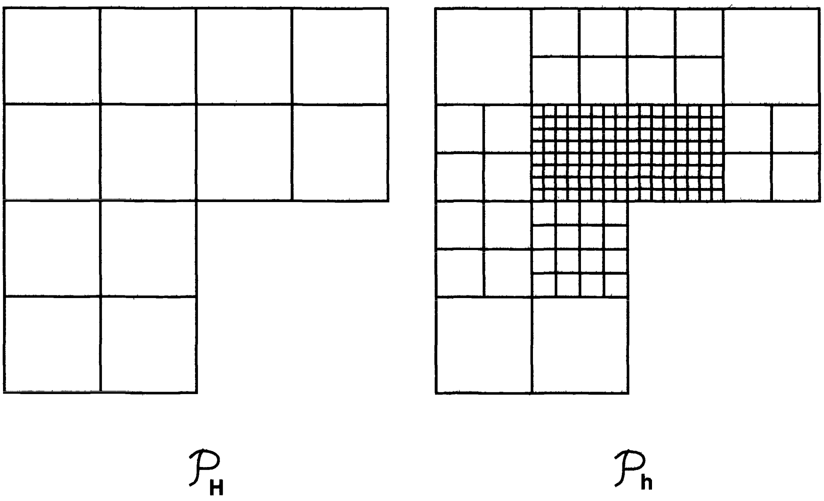


FIG. 1. *Example of a coarse mesh and locally refined mesh.*

TABLE 1
Condition number of diagonally scaled stiffness matrix for Laplacian.

	Elements in each direction (h^{-1})			
Degree (p)	1	2	4	8
1	4.73	23.88	122.07	598.63
2	22.08	49.78	227.79	1056.04
4	69.56	90.85	377.07	1755.91

well, the polynomial degree may be increased selectively. The details needed to implement selective polynomial enrichment are not trivial but are similar to the techniques described in Demkowicz et al. [8].

We shall suppose that by following some adaptive strategy, a refined partition $\mathcal{P}_h$ has been generated satisfying the above rules. The partition consists of elements from the original coarse partitioning $\mathcal{P}_H$ along with elements obtained by a locally uniform subdivision of an element from the coarse partitioning (see Figure 1).

1.4. Preconditioning. Suppose that mesh refinements and polynomial enrichments have been performed and let the associated finite element subspace be denoted by X. The finite element approximation on the refined subspace X is defined by

$$\text{Find } u \in X \text{ such that } B(u, v) = (f, v) \text{ for all } v \in X. \tag{9}$$

The discrete form of this problem is then

$$B\mathbf{x} = \mathbf{f} \tag{10}$$

where B is a symmetric positive definite matrix. The basic approach for solving the matrix equation (10) will be the conjugate gradient method. The condition number κ governs the performance of the conjugate gradient solution routine with each iteration reducing the error by at least a factor $(\sqrt{\kappa} - 1)/(\sqrt{\kappa} + 1)$. Unfortunately, it is generally found that the condition number grows rapidly as the mesh is refined or the polynomial degree increased (see Table 1). One possibility is to apply a preconditioner to the linear system. A preconditioning form

$C(\cdot,\cdot)$ is constructed for which there exist μ, Υ such that

$$\mu C(v,v) \leq B(v,v) \leq \Upsilon C(v,v) \quad \text{for all } v \in X \tag{11}$$

where μ and Υ depend on the mesh and the polynomial degree but are independent of the function v. The preconditioning form should be chosen with two properties in mind. First, the problem

$$\text{Find } w \in X \text{ such that } C(w,v) = g(v) \text{ for all } v \in X \tag{12}$$

where $g(\cdot)$ is an appropriate linear functional on X must be capable of being solved efficiently. In particular, the work required in applying the preconditioner should be modest in comparison with simply solving the problem (9) directly. Second, since the rate of convergence for the preconditioned algorithm is controlled by Υ/μ, this ratio must be controlled as the mesh is refined and the polynomial degree increased.

2. Numerical algorithm.

2.1. The basic procedure. The elements Ω_K constructed during the initial coarse partitioning $\mathcal{P}_H$ will be referred to as *subdomains*. As remarked earlier, the unrefined subdomains exist as elements in the final partitioning $\mathcal{P}_h$. The refined subdomains serve to group the elements in the final partitioning into uniform sets of elements linked by a simple topology. The preconditioner will be based on this natural domain decomposition. The overall data structure can also exploit this situation.

Step 1. Coarse Grid
- Construct the initial coarse partitioning $\mathcal{P}_H$.
- Assemble and solve the finite element problem (8) on the coarse mesh.
- Use the coarse grid solution to design the partitioning $\mathcal{P}_h$ and the space X.

Step 2. Assembly of Schur Complement
- Assemble the contributions to the global stiffness matrix and global load vector from the subdomain Ω_K.
- Apply static condensation to the subdomain stiffness matrix and subdomain load vector to eliminate the internal degrees of freedom on the subdomain.
- Assemble the reduced matrix and vector into a global matrix and vector thereby obtaining a system of the form $S\mathbf{x} = \mathbf{b}$. The Schur complement is assembled by the standard finite element subassembly (of the local Schur complements).

Step 3. Solve Schur complement $S\mathbf{x} = \mathbf{b}$
- Apply preconditioned conjugate gradient with preconditioner described below.

Step 4. Internal Degrees of Freedom
- Apply back substitution to obtain the values of the interiors of the subdomains.

One advantage of this approach is that the internal unknowns are eliminated at a local level thereby simplifying the data structure. A second advantage is that the problems associated with the interiors are constructed on a grid with regular topology.

2.2. The preconditioner. The preconditioner is based on a decomposition of the degrees of freedom (DOFs) on the edges of the subdomains into three sets:

$$\mathcal{J}_H = \{\text{DOFs associated with the vertices of the coarse grid } \mathcal{P}_H\}, \tag{13}$$

$$\mathcal{J}_h = \{\text{DOFs associated with linear functions introduced by refinement of } \mathcal{P}_H\}, \tag{14}$$

$$\mathcal{J}_p = \{\text{DOFs associated with higher-order functions}\}. \tag{15}$$

For ease of exposition, suppose that the DOFs have been ordered so that the set $\mathcal{J}_H$ is numbered first, followed by $\mathcal{J}_h$, and finally $\mathcal{J}_p$. The Schur complement and reduced load vector may be partitioned into blocks corresponding to this ordering:

$$(16) \qquad S = \begin{bmatrix} S_{HH} & S_{Hh} & S_{Hp} \\ S^t_{Hh} & S_{hh} & S_{hp} \\ S^t_{Hp} & S^t_{hp} & S_{pp} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \mathbf{b}_H \\ \mathbf{b}_h \\ \mathbf{b}_p \end{bmatrix}.$$

The solution of the preconditioning problem

$$(17) \qquad C\mathbf{x} = \mathbf{b}$$

consists of three main steps.

Linear DOFs. The DOFs within the set $\mathcal{J}_h$ are partitioned further into subsets corresponding to the separate edges $E_1, \dots, E_n$ of the coarse mesh $\mathcal{P}_H$. The submatrix S_{hh} then has the block structure

$$(18) \qquad S_{hh} = \begin{bmatrix} S^{(1,1)}_{hh} & \cdots & S^{(1,n)}_{hh} \\ \vdots & \ddots & \vdots \\ S^{(1,n)^t}_{hh} & \cdots & S^{(n,n)}_{hh} \end{bmatrix}.$$

Let

$$(19) \qquad D_{hh} = \operatorname{diag}(S^{(1,1)}_{hh}, \dots, S^{(n,n)}_{hh}).$$

Then inverting D_{hh} to obtain

$$(20) \qquad \mathbf{x}_h = D^{-1}_{hh}\mathbf{b}_h$$

corresponds to solving independent problems over each edge of the coarse mesh $\mathcal{P}_H$.

Higher-order DOFs. Similarly, dividing the higher-order DOFs in the set $\mathcal{J}_p$ into sets corresponding to each of the separate edges in the fine mesh $\mathcal{P}_h$ gives the matrix S_{pp} a block structure. Inverting the block diagonal D_{pp} of the matrix S_{pp} corresponds to solving independent problems over each edge of the fine mesh to compute

$$(21) \qquad \mathbf{x}_p = D^{-1}_{pp}\mathbf{b}_p.$$

Coarse grid. The problem associated with the coarse grid involves inverting the same stiffness matrix B_{HH} arising from the original bilinear discretization on the coarse grid $\mathcal{P}_H$. First, it is necessary to construct a *restriction* of the data $\mathbf{b}_H$ and $\mathbf{b}_h$ to the space X_H. This transformation is affected by writing the bilinear functions on the original coarse mesh as linear combinations of the linear functions on the edges of the refined mesh. That is, on the edges of $\mathcal{P}_H$ one has for a suitable constant matrix R

$$(22) \qquad \psi_H = \phi_H + R\phi_h$$

where ψ_H are the bilinear functions on the coarse grid $\mathcal{P}_H$, and ϕ_H and ϕ_h are the functions corresponding to the DOFs in the sets $\mathcal{J}_H$ and $\mathcal{J}_h$, respectively. The matrix R can be given explicitly if we let $\mathbf{x}_j$ denote the node on the edge of $\mathcal{P}_h$ at which the jth linear DOF $\phi_{h,j} \in \mathcal{J}_h$ is based; then

$$(23) \qquad R = [\,\psi_H(\mathbf{x}_j)]_{j\in\mathcal{J}_h}.$$

In practice, it is unnecessary to assemble the matrix explicitly. The relation (23) does not generally hold on the subdomain interiors. The matrix R is unchanged even if higher-order approximation is used provided that the functions $\phi_{h,j}$ are bilinear (as is the case when using hierarchical element basis functions).

The steps to construct the coarse grid correction may be summarized as follows:

- Form the *restriction* of the data
$$\hat{\mathbf{b}}_H = \mathbf{b}_H + R\mathbf{b}_h.$$
- Solve the coarse grid problem
$$B_{HH}\hat{\mathbf{x}}_H = \hat{\mathbf{b}}_H$$
where B_{HH} is the stiffness matrix for the discretization (8).
- *Prolongate* the solution $\hat{\mathbf{x}}_H$ to the space spanned by elements of $\mathcal{J}_H$ and $\mathcal{J}_h$ according to
$$\mathbf{x}_H^c = \hat{\mathbf{x}}_H, \quad \mathbf{x}_h^c = R^t\hat{\mathbf{x}}_H.$$

The solution of the preconditioning problem (17) is then given by

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_H^c \\ \mathbf{x}_h + \mathbf{x}_h^c \\ \mathbf{x}_p \end{bmatrix}. \tag{24}$$

The action of the inverse of the preconditioner C therefore corresponds to solving independent problems for the linear DOFs over each of the edges of $\mathcal{P}_H$ separately, independent solves for the higher-order DOFs over each of the edges of $\mathcal{P}_h$ separately, and a global solve over the DOFs in the coarse grid discretization. Moreover, each of these overall steps may be performed concurrently. In particular, the computations are ideally suited to a parallel programming environment.

As remarked earlier, the performance of the preconditioner is governed by the quantities μ and Υ in the equivalence

$$\mu C(v, v) \le B(v, v) \le \Upsilon C(v, v) \text{ for all } v \in X. \tag{25}$$

The algorithm presented above generalizes both an algorithm proposed by Smith [13] for the h-version finite element method and an algorithm proposed by Mandel [11] for the p-version. For the case of first-order ($p = 1$) approximation, Smith [13] has shown that the ratio grows at most as

$$\frac{\Upsilon}{\mu} \le C\left(1 + \log^2\left(\frac{H}{h}\right)\right) \tag{26}$$

where the constant C is independent of H and h. Conversely, for the p-version finite element method ($H/h = 1$), it has been shown [2] that the algorithm given by Mandel [11] yields

$$\frac{\Upsilon}{\mu} \le C(1 + \log^2 p) \tag{27}$$

where the constant C is independent of p and the number of elements. It is important to note that, in both cases, the constant C is independent of the variation of the coefficients a_{jk} between subdomains. The purpose of the present work is to obtain estimates of this form for the combined h-p version of the finite element method.

3. Analysis of preconditioner.

3.1. Algebraic formulation. Suppose that the standard basis functions are partitioned into the set consisting of functions $\{\phi_i\}$, $i \in \mathcal{I}$ supported on the interiors of the subdomains and the set of functions $\{\phi_i\}$, $i \in \tilde{\mathcal{I}}$ having nonzero values on at least one of the subdomain interfaces $\partial\mathcal{P}_H$. Steps 1 and 2 of the algorithm involved the static condensation of the internal DOFs on each of the subdomains. This process may be interpreted as a change of basis. Specifically, the edge basis functions $\{\phi_i\}$, $i \in \tilde{\mathcal{I}}$ are replaced by new functions $\{\tilde{\phi}_i\}$, $i \in \tilde{\mathcal{I}}$ satisfying

$$\tilde{\phi}_i = \phi_i \quad \text{on } \partial\mathcal{P}_H \tag{28}$$

and

$$B(\tilde{\phi}_i, \phi_j) = 0 \quad \text{for all } j \in \mathcal{I}. \tag{29}$$

The modified edge functions $\tilde{\phi}_i$, $i \in \tilde{\mathcal{I}}$ are obtained from the original edge functions by subtracting linear combinations of the internal functions. Introducing vectors defined by

$$\phi_{\mathcal{I}} = [\phi_i]_{i\in\mathcal{I}}, \quad \phi_{\tilde{\mathcal{I}}} = [\phi_i]_{i\in\tilde{\mathcal{I}}}, \quad \tilde{\phi}_{\tilde{\mathcal{I}}} = [\tilde{\phi}_i]_{i\in\tilde{\mathcal{I}}}, \tag{30}$$

then, for a suitable matrix T, we may write the change of basis as

$$\begin{bmatrix} \tilde{\phi}_{\tilde{\mathcal{I}}} \\ \phi_{\mathcal{I}} \end{bmatrix} = \begin{bmatrix} I & -T \\ 0 & I \end{bmatrix} \begin{bmatrix} \phi_{\tilde{\mathcal{I}}} \\ \phi_{\mathcal{I}} \end{bmatrix}. \tag{31}$$

The matrix T can be obtained by writing the stiffness matrix for the problem (10) in a block form corresponding to the ordering of the basis functions into the sets $\mathcal{I}$ and $\tilde{\mathcal{I}}$:

$$B = \begin{bmatrix} B_{\mathcal{I}\mathcal{I}} & B_{\mathcal{I}\tilde{\mathcal{I}}} \\ B_{\tilde{\mathcal{I}}\mathcal{I}} & B_{\tilde{\mathcal{I}}\tilde{\mathcal{I}}} \end{bmatrix}; \tag{32}$$

then condition (29) yields

$$T = B_{\tilde{\mathcal{I}}\mathcal{I}} B_{\mathcal{I}\mathcal{I}}^{-1}. \tag{33}$$

The stiffness matrix for the problem relative to the transformed basis would then be

$$\tilde{B} = \begin{bmatrix} I & -T \\ 0 & I \end{bmatrix} \begin{bmatrix} B_{\mathcal{I}\mathcal{I}} & B_{\mathcal{I}\tilde{\mathcal{I}}} \\ B_{\tilde{\mathcal{I}}\mathcal{I}} & B_{\tilde{\mathcal{I}}\tilde{\mathcal{I}}} \end{bmatrix} \begin{bmatrix} I & -T \\ 0 & I \end{bmatrix}^t = \begin{bmatrix} S & 0 \\ 0 & B_{\mathcal{I}\mathcal{I}} \end{bmatrix} \tag{34}$$

where the matrix S is the Schur complement (16)

$$S = B_{\tilde{\mathcal{I}}\tilde{\mathcal{I}}} - B_{\tilde{\mathcal{I}}\mathcal{I}} B_{\mathcal{I}\mathcal{I}}^{-1} B_{\mathcal{I}\tilde{\mathcal{I}}}. \tag{35}$$

The original problem (10) written in block form is

$$B \begin{bmatrix} \mathbf{x}_{\tilde{\mathcal{I}}} \\ \mathbf{x}_{\mathcal{I}} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_{\tilde{\mathcal{I}}} \\ \mathbf{f}_{\mathcal{I}} \end{bmatrix}. \tag{36}$$

Applying the transformation (31) gives

$$\begin{bmatrix} S & 0 \\ B_{\mathcal{I}\tilde{\mathcal{I}}} & B_{\mathcal{I}\mathcal{I}} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{\tilde{\mathcal{I}}} \\ \mathbf{x}_{\mathcal{I}} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_{\tilde{\mathcal{I}}} - B_{\tilde{\mathcal{I}}\mathcal{I}} B_{\mathcal{I}\mathcal{I}}^{-1} \mathbf{f}_{\mathcal{I}} \\ \mathbf{f}_{\mathcal{I}} \end{bmatrix}. \tag{37}$$

This problem is solved in two stages (cf. Steps 3 and 4):

$$S\mathbf{x}_{\tilde{\mathcal{I}}} = \mathbf{f}_{\tilde{\mathcal{I}}} - B_{\tilde{\mathcal{I}}\mathcal{I}}B_{\mathcal{I}\mathcal{I}}^{-1}\mathbf{f}_{\mathcal{I}} \tag{38}$$

and

$$B_{\mathcal{I}\mathcal{I}}\mathbf{x}_{\mathcal{I}} = \mathbf{f}_{\mathcal{I}} - B_{\mathcal{I}\tilde{\mathcal{I}}}\mathbf{x}_{\tilde{\mathcal{I}}}. \tag{39}$$

The preconditioner is then applied to the reduced Schur complement system (38). Let $X_0 \in X$ be the space

$$X_0 = \operatorname{span}\left\{\tilde{\phi}_i : \quad i \in \tilde{\mathcal{I}}\right\}. \tag{40}$$

In view of condition (29), elements of the space X_0 are referred to as *discrete harmonic* functions. The Schur complement matrix S induces a bilinear form $S(\cdot,\cdot)$ on the space X_0 which agrees with the original bilinear form $B(\cdot,\cdot)$.

LEMMA 3.1. *Let u, v be discrete harmonic. Then*

$$S(u,v) = B(u,v). \tag{41}$$

Proof. Let $w \in X_0$. Then w may be written as

$$w = \alpha^t_{\tilde{\mathcal{I}}}\tilde{\phi}_{\tilde{\mathcal{I}}}.$$

Applying (34) gives

$$B(w,w) = \begin{bmatrix} \alpha_{\tilde{\mathcal{I}}} \\ 0 \end{bmatrix}^t \begin{bmatrix} S & 0 \\ 0 & B_{\mathcal{I}\mathcal{I}} \end{bmatrix} \begin{bmatrix} \alpha_{\tilde{\mathcal{I}}} \\ 0 \end{bmatrix} = \alpha^t_{\tilde{\mathcal{I}}} S\, \alpha_{\tilde{\mathcal{I}}} = S(w,w).$$

Applying this result to $w = u + v$ and simplifying completes the proof. □

The preconditioner was constructed by decomposing the degrees of freedom on the edges of the subdomains into three sets: $\mathcal{J}_H$, $\mathcal{J}_h$, and $\mathcal{J}_p$. The space X_0 is decomposed into subspaces X_0^H, X_0^h, and X_0^p using this partitioning where

$$X_0^H = \operatorname{span}\{\tilde{\phi}_i, i \in \mathcal{J}_H\} \tag{42}$$

and X_0^h and X_0^p are defined similarly. The matrix R defined in (23) maps X_0^h into X_0^H. An alternative viewpoint is to regard R as another change of basis from $\tilde{\phi}_{\tilde{\mathcal{I}}}$ to $\tilde{\psi}$ given by

$$\begin{bmatrix} \tilde{\psi}^H \\ \tilde{\psi}^h \\ \tilde{\psi}^p \end{bmatrix} = Q \begin{bmatrix} \phi^H_{\tilde{\mathcal{I}}} \\ \phi^h_{\tilde{\mathcal{I}}} \\ \phi^p_{\tilde{\mathcal{I}}} \end{bmatrix} \tag{43}$$

where

$$Q = \begin{bmatrix} I & R & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix}. \tag{44}$$

It is easily seen that the inverse of the matrix Q is given by

$$Q^{-1} = \begin{bmatrix} I & -R & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix}. \tag{45}$$

The Schur complement matrix S relative to the new basis has the form $\widehat{S}$:

$$\widehat{S} = QSQ^t = \begin{bmatrix} \widehat{S}_{HH} & X & X \\ X & S_{hh} & X \\ X & X & S_{pp} \end{bmatrix} \tag{46}$$

where X denotes appropriate matrices,

$$\widehat{S}_{HH} = \begin{bmatrix} I & R \end{bmatrix} \begin{bmatrix} S_{HH} & S_{Hh} \\ S_{hH} & S_{hh} \end{bmatrix} \begin{bmatrix} I \\ R^t \end{bmatrix}, \tag{47}$$

and $S_{HH}, \ldots, S_{pp}$ are given in (16). The submatrices appearing on the main diagonal of this matrix are approximated by (see §2.2)

$$\widehat{S}_{HH} \approx B_{HH}, \quad S_{hh} \approx D_{hh}, \quad S_{pp} \approx D_{pp}, \tag{48}$$

and, equally well,

$$\widehat{S} \approx \widehat{C} = \begin{bmatrix} B_{HH} & 0 & 0 \\ 0 & D_{hh} & 0 \\ 0 & 0 & D_{pp} \end{bmatrix}. \tag{49}$$

Transforming back to the basis $\tilde{\phi}_{\mathcal{I}}$ then gives an approximation C to the original Schur complement S:

$$S \approx C = Q^{-1}\widehat{C}Q^{-t}. \tag{50}$$

Equation (50) shows that

$$C^{-1} = Q^t\widehat{C}^{-1}Q \tag{51}$$

which, on expanding, gives

$$C^{-1} = \begin{bmatrix} I \\ R^t \\ 0 \end{bmatrix} B_{HH}^{-1} \begin{bmatrix} I & R & 0 \end{bmatrix} + \begin{bmatrix} 0 \\ I \\ 0 \end{bmatrix} D_{hh}^{-1} \begin{bmatrix} 0 & I & 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ I \end{bmatrix} D_{pp}^{-1} \begin{bmatrix} 0 & 0 & I \end{bmatrix}. \tag{52}$$

The decomposition (52) shows that the action of the inverse of C consists of the same three steps described in §2.2 for the action of the inverse of the preconditioner. The matrix C in (50) is the matrix representation of the preconditioner.

3.2. Formulation as an additive Schwarz method. Introduce inner products A^H, A^h, and A^p on the spaces X_0^H, X_0^h, and X_0^p, respectively, as follows:

- For given $u \in X_0^H$, noting that u is linear on the edges of the partition $\mathcal{P}_H$, let $\Pi_H u \in X_H$ be the bilinear function which interpolates to u at the nodes of the coarse grid. Evidently, $\Pi_H u$ and u agree on the boundaries $\partial\mathcal{P}^H$. The inner product A^H is then defined by

$$A^H(u, v) = B(\Pi_H u, \Pi_H v). \tag{53}$$

- For any edge e in the coarse partition $\mathcal{P}_H$, let $E_e^h : X_0^h \mapsto X_0^h$ be defined on the edges of the coarse partition by

$$E_e^h u = \begin{cases} u & \text{on } e, \\ 0 & \text{on remaining edges in } \partial\mathcal{P}_H \end{cases} \tag{54}$$

and discrete harmonic on the interiors. Note that this operator is well defined since functions belonging to the space X_0^h vanish at the nodes of the coarse grid. For given $u \in X_0^h$, we may decompose u as

$$u = \sum_{e\in\partial\mathcal{P}^H} E_e^h u. \tag{55}$$

The inner product A^h is then defined by

$$A^h(u, v) = \sum_{e\in\partial\mathcal{P}^H} S(E_e^h u, E_e^h v). \tag{56}$$

- Similarly, for any edge $e \in \partial\mathcal{P}_h$, let $E_e^p : X_0^p \mapsto X_0^p$ be defined on $\partial\mathcal{P}_h$ by

$$E_e^p u = \begin{cases} u & \text{on } e, \\ 0 & \text{on remaining edges in } \partial\mathcal{P}_h \end{cases} \tag{57}$$

and discrete harmonic on the interiors. For $u \in X_0^p$ write

$$u = \sum_{e\in\partial\mathcal{P}^h} E_e^p u \tag{58}$$

and let the inner product A^p be given by

$$A^p(u, v) = \sum_{e\in\partial\mathcal{P}^h} S(E_e^p u, E_e^p v). \tag{59}$$

Let $P^H : X_0 \mapsto X_0^H$ denote the projection defined by

$$A^H(P^H u, v) = S(u, v) \quad \text{for all } v \in X_0^H. \tag{60}$$

Define $P^h : X_0 \mapsto X_0^h$ and $P^p : X_0 \mapsto X_0^p$ similarly.

LEMMA 3.2. *Relative to the basis $\tilde{\phi}_{\mathcal{I}}$, the projection operators P^H, P^h, and P^p have the matrix representations*

$$P^H = \begin{bmatrix} I \\ R^t \\ 0 \end{bmatrix} B_{HH}^{-1} \begin{bmatrix} I & R & 0 \end{bmatrix} S, \tag{61}$$

$$P^h = \begin{bmatrix} 0 \\ I \\ 0 \end{bmatrix} D_{hh}^{-1} \begin{bmatrix} 0 & I & 0 \end{bmatrix} S, \tag{62}$$

and

$$P^p = \begin{bmatrix} 0 \\ 0 \\ I \end{bmatrix} D_{pp}^{-1} \begin{bmatrix} 0 & 0 & I \end{bmatrix} S. \tag{63}$$

Proof. Let $u \in X_0$ be represented relative to the basis $\tilde{\psi}$ by $[\mathbf{x}^H, \mathbf{x}^h, \mathbf{x}^p]$ and let $P^H u$ be represented by $[\mathbf{z}^H, 0, 0]$. Condition (60) implies that $\mathbf{z}^H$ satisfies

$$(\mathbf{y}^H)^t B_{HH} \mathbf{z}^H = \begin{bmatrix} \mathbf{y}^H \\ 0 \\ 0 \end{bmatrix}^t \widehat{S} \begin{bmatrix} \mathbf{x}^H \\ \mathbf{x}^h \\ \mathbf{x}^p \end{bmatrix}$$

for all $\mathbf{y}^H$. Hence,

$$\mathbf{z}^H = B_{HH}^{-1} \begin{bmatrix} I & R & 0 \end{bmatrix} S Q^t \begin{bmatrix} \mathbf{x}^H \\ \mathbf{x}^h \\ \mathbf{x}^p \end{bmatrix}.$$

Transforming to the basis $\tilde{\phi}_{\tilde{\mathcal{I}}}$ gives

$$P^H u = \begin{bmatrix} I \\ R^t \\ 0 \end{bmatrix} B_{HH}^{-1} \begin{bmatrix} I & R & 0 \end{bmatrix} S u$$

from which (61) follows. The remaining results are obtained in a similar manner. □

Examining the foregoing developments, one observes that

$$C^{-1} S = P^H + P^h + P^p. \tag{64}$$

The significance of (64) is that the preconditioner developed in §2.2 can be viewed as an additive Schwarz method [7, 9, 14]. The condition number of the additive Schwarz method $\kappa(C^{-1}S)$ can be bounded using the following result (see [9, 14]).

THEOREM 3.3. *Suppose that there exist positive quantities λ and Λ such that the following conditions are valid:*

1. *For any $u \in X_0$ there exists a decomposition*

$$u = u^H + u^h + u^p \tag{65}$$

such that $u^H \in X_0^H$, $u^h \in X_0^h$, $u^p \in X_0^p$, and

$$A^H(u^H, u^H) + A^h(u^h, u^h) + A^p(u^p, u^p) \le \Lambda S(u, u). \tag{66}$$

2. *For $s \in \{H, h, p\}$ there holds*

$$S(u, u) \le \lambda^{-1} A^s(u, u) \textit{ for all } u \in X_0^s. \tag{67}$$

Then

$$\frac{1}{\Lambda} S(u, u) \le S(Pu, u) \le 3\lambda S(u, u) \tag{68}$$

where $P = P^H + P^h + P^p$. Consequently,

$$\kappa(C^{-1}S) = \kappa(P) \le 3\Lambda\lambda. \tag{69}$$

3.3. Estimation of condition number. To begin with, we assume that the fine partitioning $\mathcal{P}_h$ is obtained by uniform refinement of all the elements in the coarse partition $\mathcal{P}_H$. Moreover, it is assumed that the polynomial degree is uniform throughout the domain. Later, we shall consider the case of local h and local p refinements.

Let Ω_K be a subdomain from the coarse partition $\mathcal{P}_H$. The space $X_{0,K}$ consists of the restrictions to the subdomain Ω_K of functions belonging to X_0. The spaces $X_{0,K}^H$, $X_{0,K}^h$, and $X_{0,K}^p$ are defined analogously. Let $B_K(\cdot,\cdot)$ denote the local contribution to bilinear form $B(\cdot,\cdot)$ from the subdomain Ω_K.

Let e be any edge lying on the boundary $\partial\Omega_K$ The space $H_{00}^{1/2}(e)$ consists of functions defined on the edge $e \subset \partial\Omega_K$ which have continuous extensions by zero to the rest of the boundary $\partial\Omega_K$ [10]. A norm may be defined on this space by

$$(70)\qquad |v|^2_{H_{00}^{1/2}(e)} = \int_e\int_e \frac{|v(\mathbf{x})-v(\mathbf{y})|^2}{|\mathbf{x}-\mathbf{y}|^2}\,\mathrm{d}s_x\,\mathrm{d}s_y + \int_e \left\{ \frac{|v(\mathbf{x})|^2}{|\mathbf{x}-\mathbf{x}_{e_1}|} + \frac{|v(\mathbf{x})|^2}{|\mathbf{x}-\mathbf{x}_{e_2}|} \right\} \mathrm{d}s_x$$

where $\mathbf{x}_{e_1}$ and $\mathbf{x}_{e_2}$ are the endpoints of the edge e. Finally, Π_h denotes the piecewise linear interpolation operator at the nodes of the fine grid $\mathcal{P}_h$. It will be useful to recall the following results proved in [1].

LEMMA 3.4. *Let $u \in X_{0,K}$. Then there exists a constant C independent of H, h, p, and u such that the following hold:*

1.

$$(71)\qquad \sum_{e\in\partial K\cap\partial\mathcal{P}_H} \left(u(\mathbf{x}_{e_1}) - u(\mathbf{x}_{e_2})\right)^2 \le C\left(1+\log\left(\frac{Hp}{h}\right)\right) B_K(u,u).$$

2.

$$(72)\qquad \sum_{e\in\partial K\cap\partial\mathcal{P}_H} |\Pi_h u - \Pi_H u|^2_{H_{00}^{1/2}(e)} \le C\left(1+\log\left(\frac{Hp}{h}\right)\right)^2 B_K(u,u).$$

3.

$$(73)\qquad \sum_{e\in\partial K\cap\partial\mathcal{P}_h} |u - \Pi_h u|^2_{H_{00}^{1/2}(e)} \le C\,(1+\log p)^2\, B_K(u,u).$$

4. *If $u \in X_{0,K}^p$ then*

$$(74)\qquad B_K(u,u) \le C \sum_{e\in\partial K\cap\partial\mathcal{P}_h} |u|^2_{H_{00}^{1/2}(e)}\,.$$

Proof. See Ainsworth [1]. □

LEMMA 3.5. *There exist constants C independent of H, h, p, and u such that the following hold:*

1. *If $u \in X_{0,K}^H$ then*

$$(75)\qquad B_K(u,u) \le B_K(\Pi_H u, \Pi_H u).$$

2. *If $u \in X_{0,K}^h$ then*

$$(76)\qquad B_K(u,u) \le C \sum_{e\in\partial K\cap\partial\mathcal{P}_H} B_K(E_e^h u, E_e^h u).$$

3. *If* $u \in X^p_{0,K}$ *then*

$$B_K(u,u) \le C \min\left(\frac{H}{h}, 1+\log^2 p\right) \sum_{e\in\partial K\cap\partial\mathcal{P}_h} B_K(E^p_e u, E^p_e u). \tag{77}$$

Proof.

1. Let $\Pi_H u \in X_H$. Then on the subdomain Ω_K

$$\Pi_H u = u + u_{\mathcal{I}}$$

where $u_{\mathcal{I}}$ is supported on Ω_K. Then, by condition (29),

$$\begin{aligned} B_K(\Pi_H u, \Pi_H u) &= B_K(u,u) + 2B(u,u_{\mathcal{I}}) + B_K(u_{\mathcal{I}}, u_{\mathcal{I}}) \\ &= B_K(u,u) + B_K(u_{\mathcal{I}}, u_{\mathcal{I}}) \\ &\ge B_K(u,u). \end{aligned}$$

2. Let $u \in X^h_{0,K}$ be written on the subdomain in the form

$$u = \sum_{e\in\partial K\cap\partial\mathcal{P}_H} E^h_e u.$$

Substituting for u and using the Cauchy–Schwarz inequality, we have

$$B_K(u,u) \le \sum_{e\in\partial K\cap\partial\mathcal{P}_H} 1 \cdot \sum_{e\in\partial K\cap\partial\mathcal{P}_H} B_K(E^h_e u, E^h_e u) \le C \sum_{e\in\partial K\cap\partial\mathcal{P}_H} B_K(E^h_e u, E^h_e u).$$

3. Let $u \in X^p_{0,K}$. Decompose u on the subdomain in the form

$$u = \sum_{e\in\partial K\cap\partial\mathcal{P}_h} E^p_e u.$$

Then, applying Lemma 3.4(4),

$$\begin{aligned} B_K(u,u) &\le C \sum_{e\in\partial K\cap\partial\mathcal{P}_h} |u|^2_{H^{1/2}_{00}(e)} \\ &= C \sum_{e\in\partial K\cap\partial\mathcal{P}_h} \left|E^p_e u\right|^2_{H^{1/2}_{00}(e)} \\ &\le C\,(1+\log p)^2 \sum_{e\in\partial K\cap\partial\mathcal{P}_h} B_K(E^p_e u, E^p_e u) \end{aligned}$$

where Lemma 3.4(3) has been applied to each of the functions $E^p_e u$ separately. Alternatively, observing that there are H/h elements on each subdomain edge, the Cauchy–Schwarz inequality implies

$$\begin{aligned} B_K(u,u) &\le C \sum_{e\in\partial K\cap\partial\mathcal{P}_h} 1 \cdot \sum_{e\in\partial K\cap\partial\mathcal{P}_h} B_K(E^p_e u, E^p_e) \\ &\le C\left(\frac{H}{h}\right) \sum_{e\in\partial K\cap\partial\mathcal{P}_h} B_K(E^p_e u, E^p_e). \end{aligned}$$

This completes the proof. □

LEMMA 3.6. *Let* $u \in X_{0,K}$. *Then there exists a decomposition of the form*

$$u = u^H + u^h + u^p \tag{78}$$

where $u^H \in X^H_{0,K}$, $u^h \in X^h_{0,K}$, *and* $u^p \in X^p_{0,K}$ *are such that*

$$B_K(\Pi_H u^H, \Pi_H u^H) + \sum_{e \in \partial K \cap \partial \mathcal{P}_H} B_K(E^h_e u^h, E^h_e u^h) + \sum_{e \in \partial K \cap \partial \mathcal{P}_h} B_K(E^p_e u^p, E^p_e u^p)$$
$$\le C\left(1 + \log\left(\frac{Hp}{h}\right)\right)^2 B_K(u,u) \tag{79}$$

where C is independent of H, h, p, and u.

Proof. Let $u \in X_{0,K}$. Choose $u^H \in X^H_{0,K}$, $u^h \in X^h_{0,K}$, and $u^p \in X^p_{0,K}$ such that on the subdomain boundary $\partial\Omega_K$ there holds

$$u^H = \Pi_H u, \quad u^h = \Pi_h u - \Pi_H u, \quad u^p = u - \Pi_h u.$$

Then, thanks to the quasiuniformity of the coarse grid $\mathcal{P}_H$ and Lemma 3.4(1),

$$\begin{aligned} B_K(\Pi_H u^H, \Pi_H u^H) &= B_K(\Pi_H u, \Pi_H u) \\ &\le C \sum_{e \in \partial K \cap \partial \mathcal{P}_H} \left(u(\mathbf{x}_{e_1}) - u(\mathbf{x}_{e_2})\right)^2 \\ &\le \left(1 + \log\left(\frac{Hp}{h}\right)\right) B_K(u,u) \end{aligned} \tag{80}$$

where $\mathbf{x}_{e_1}$ and $\mathbf{x}_{e_2}$ are the endpoints of the edge $e \in \partial\mathcal{P}_H$. Using Lemma 3.4(2)

$$\begin{aligned} \sum_{e \in \partial K \cap \partial \mathcal{P}_H} B_K(E^h_e u^h, E^h_e u^h) &\le C \sum_{e \in \partial K \cap \partial \mathcal{P}_H} |E^h_e u^h|^2_{H^{1/2}_{00}(e)} \\ &= C \sum_{e \in \partial K \cap \partial \mathcal{P}_H} |\Pi_H u - \Pi_h u|^2_{H^{1/2}_{00}(e)} \\ &\le C\left(1 + \log\left(\frac{Hp}{h}\right)\right)^2 B_K(u,u). \end{aligned} \tag{81}$$

Using Lemma 3.4(4)

$$\begin{aligned} \sum_{e \in \partial K \cap \partial \mathcal{P}_h} B_K(E^p_e u^p, E^p_e u^p) &\le C \sum_{e \in \partial K \cap \partial \mathcal{P}_h} |E^p_e u^p|^2_{H^{1/2}_{00}(e)} \\ &= C \sum_{e \in \partial K \cap \partial \mathcal{P}_h} |u - \Pi_h u|^2_{H^{1/2}_{00}(e)} \\ &\le C\left(1 + \log\left(\frac{Hp}{h}\right)\right)^2 B_K(u,u). \end{aligned} \tag{82}$$

The proof is completed by summing (80)–(82). □

THEOREM 3.7. *The condition number of the preconditioned system* $\kappa(C^{-1}S)$ *is bounded by*

$$\kappa(C^{-1}S) \le \frac{3K_1}{K_2}\left(1 + \log\left(\frac{Hp}{h}\right)\right)^2 \min\left(\frac{H}{h}, 1 + \log^2 p\right). \tag{83}$$

The constants K_1 *and* K_2 *are independent of H, h, and p. Moreover, the constants depend only on the variation of the coefficients* a_{jk} *within each of the subdomains. Consequently, the ratio* K_1/K_2 *is independent of the magnitude of the jump discontinuities of the coefficients between neighbouring subdomains.*

Proof. The result follows after checking the conditions of Theorem 3.3. First, let $u \in X_0$ be decomposed as

$$u = \sum_{K \in \mathcal{P}_H} u_K$$

where $u_K \in X_0^K$. Each of the functions u_K is then decomposed using the splitting of Lemma 3.6:

$$u_K = u_K^H + u_K^h + u_K^p.$$

Thanks to the definition of the functions u_K^H, u_K^h, and u_K^p on the boundaries $\partial\Omega_K$ we may construct a decomposition of u as follows:

$$u = u^H + u^h + u^p$$

with $u^H \in X_0^H$, $u^h \in X_0^h$, and $u^p \in X_0^p$ defined as

$$u^H = \sum_{K \in \mathcal{P}_H} u_K^H, \quad u^h = \sum_{K \in \mathcal{P}_H} u_K^h, \quad u^p = \sum_{K \in \mathcal{P}_H} u_K^p.$$

It then follows that

$$\begin{aligned}
&A^H(u^H, u^H) + A^h(u^h, u^h) + A^p(u^p, u^p) \\
&= B(\Pi_H u^H, \Pi_H u^H) + \sum_{e \in \partial\mathcal{P}_H} S(E_e^h u^h, E_e^h u^h) + \sum_{e \in \partial\mathcal{P}_h} S(E_e^p u^p, E_e^p u^p) \\
&= B(\Pi_H u^H, \Pi_H u^H) + \sum_{e \in \partial\mathcal{P}_H} B(E_e^h u^h, E_e^h u^h) + \sum_{e \in \partial\mathcal{P}_h} B(E_e^p u^p, E_e^p u^p) \quad \text{(Lemma 3.1)} \\
&= \sum_{K \in \mathcal{P}_H} \left\{ B_K(\Pi_H u^H, \Pi_H u^H) + \sum_{e \in \partial\mathcal{P}_H \cap \partial K} B_K(E_e^h u^h, E_e^h u^h) + \sum_{e \in \partial\mathcal{P}_h \cap \partial K} B_K(E_e^p u^p, E_e^p u^p) \right\} \\
&\le \sum_{K \in \mathcal{P}_H} C_K \left(1 + \log\left(\frac{Hp}{h}\right)\right)^2 B_K(u, u) \quad \text{(Lemma 3.6)} \\
&\le \left(\max_{K \in \mathcal{P}_H} C_K\right) \left(1 + \log\left(\frac{Hp}{h}\right)\right)^2 B(u, u) \\
&\le \left(\max_{K \in \mathcal{P}_H} C_K\right) \left(1 + \log\left(\frac{Hp}{h}\right)\right)^2 S(u, u) \quad \text{(Lemma 3.1)}
\end{aligned}$$

so that the first condition is verified with $\Lambda = \left(\max_{K \in \mathcal{P}_H} C_K\right)(1 + \log(Hp/h))^2$. Turning now to the second set of conditions, suppose $u \in X_0^H$. By Lemmas 3.1 and 3.5(1)

$$S(u, u) = \sum_{K \in \mathcal{P}_H} B_K(u, u) \le \sum_{K \in \mathcal{P}_H} B_K(\Pi_H u, \Pi_H u) = B(\Pi_H u, \Pi_H u) = A^H(u, u). \tag{84}$$

Let $u \in X_0^h$. Then, using Lemmas 3.1 and 3.5(2),

$$\begin{aligned}
S(u, u) &= \sum_{K \in \mathcal{P}_H} B_K(u, u) \\
&\le c_K \sum_{K \in \mathcal{P}_H} \sum_{e \in \partial\mathcal{P}_H \cap \partial K} B_K(E_e^h u, E_e^h u) \\
&\le c_K \sum_{e \in \partial\mathcal{P}_H} B(E_e^h u, E_e^h u) = c_K \sum_{e \in \partial\mathcal{P}_H} S(E_e^h u, E_e^h u) = c_K A^h(u, u). \qquad (85)
\end{aligned}$$

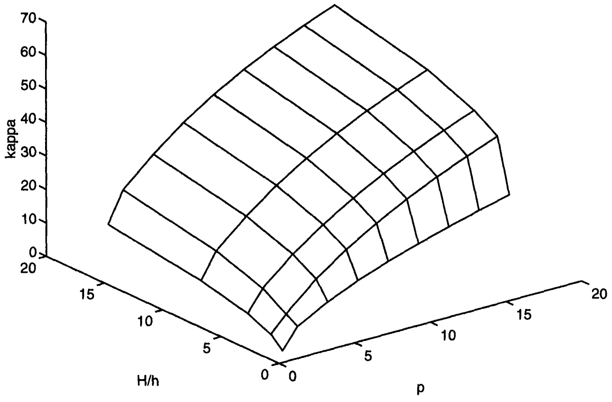


FIG. 2. *Computed bounds on condition number for preconditioner based on $\widehat{B}_{HH}$.*

Similarly, applying Lemmas 3.1 and 3.5(3) gives, for $u \in X_0^p$,

$$S(u,u) \le c_K \min\left(\frac{H}{h}, 1+\log^2 p\right) A^p(u,u). \tag{86}$$

Collecting inequalities (84)–(86) shows that the second condition of Theorem 3.3 is valid with λ given by $\left(\min_{K\in\mathcal{P}_H} c_K\right)\min\left(H/h, 1+\log^2 p\right)$. □

3.4. Numerically computed condition numbers. Let K be a square subdomain and $B_K(\cdot,\cdot)$ be the bilinear form corresponding to the Laplace operator over the subdomain K:

$$B_K(u,v) = \int_K \nabla u \cdot \nabla v \,\mathbf{dx}. \tag{87}$$

In order to assess the performance of the preconditioner in practice we compute estimates of the condition number. The numerical computations will be based on the following result.

LEMMA 3.8. *Let $S_K(\cdot,\cdot)$ and $C_K(\cdot,\cdot)$ denote the Schur complement and preconditioner restricted to the subspace X_0^K. Suppose that there exist quantities μ_K and Υ_K such that for all $u \in X_0^K$*

$$\mu_K \le \frac{S_K(u,u)}{C_K(u,u)} \le \Upsilon_K. \tag{88}$$

Then

$$\kappa(C^{-1}S) \le \frac{\Upsilon}{\mu} \tag{89}$$

where $\Upsilon = \max_K \Upsilon_K$ and $\mu = \min_K \mu_K$.

Proof. Sum (88) over all subdomains $K \in \mathcal{P}_H$. □

This result shows that one may obtain estimates for the performance of the preconditioner when applied to a partition $\mathcal{P}_H$ consisting of uniform subdomains Ω_K. Figure 2 and Table 2 show the computed condition numbers for the preconditioner described in §2. Table 2 also shows that for large values of p and H/h, the condition number can be approximated by

$$\tilde{\kappa} = 13.3493\left(1 - 0.0050\log\left(\frac{Hp}{h}\right) + 0.0280\log^2\left(\frac{Hp}{h}\right)\right) \cdot \left(1 + 0.2803\log p + 0.0980\log^2 p\right) \tag{90}$$

in agreement with the main conclusion of Theorem 3.7.

TABLE 2

Computed bounds on condition number for preconditioner based on $\widehat{B}_{HH}$. (Numbers in parentheses show the values obtained when the data is fitted using the approximation $\tilde{\kappa}$.)

	Elements in each direction (H/h)				
Degree (p)	1	2	4	8	16
1	1.00	4.23	7.28	10.72	14.73
	(13.35)	(13.48)	(13.98)	(14.83)	(16.04)
2	6.99	9.11	13.59	18.42	23.78
	(16.74)	(17.35)	(18.40)	(19.91)	(21.85)
4	11.83	17.10	21.32	26.33	31.98
	(22.04)	(23.38)	(25.29)	(27.76)	(30.80)
6	15.75	23.47	27.74	33.04	38.99
	(26.22)	(28.14)	(30.72)	(33.95)	(37.84)
8	19.05	28.89	33.17	38.68	44.85
	(29.75)	(32.18)	(35.33)	(39.19)	(43.78)
10	21.91	33.61	37.98	43.57	49.86
	(32.86)	(35.73)	(39.37)	(43.80)	(48.99)
12	24.52	37.80	42.22	47.88	54.24
	(35.65)	(38.92)	(43.01)	(47.93)	(53.68)
14	26.92	41.57	46.03	51.75	58.16
	(38.21)	(41.84)	(46.34)	(51.71)	(57.95)
16	29.11	45.01	49.49	55.25	61.70
	(40.58)	(44.55)	(49.43)	(55.21)	(61.90)

TABLE 3

Computed bounds on condition number for preconditioner based on $\widehat{S}_{HH}$.

	Elements in each direction (H/h)				
Degree (p)	1	2	4	8	16
1	1.00	4.23	7.28	10.72	14.73
2	6.99	9.11	13.59	18.42	23.78
4	11.83	15.66	19.67	23.65	28.30
6	15.75	21.39	25.42	29.03	33.18
8	19.05	26.34	30.54	33.85	37.60
10	21.91	30.67	35.04	38.09	41.60
12	24.52	34.54	39.04	41.89	45.18
14	26.92	38.03	42.67	45.34	48.45
16	29.11	41.23	45.99	48.50	51.45

The discussion in §3.1 suggests that a better condition number will be obtained by using the matrix $\widehat{S}_{HH}$ defined in (47) instead of the coarse grid discretization matrix B_{HH}. The results in Table 3 confirm that the condition number is reduced slightly.

3.5. Local refinements. Suppose now that the fine grid $\mathcal{P}_h$ is obtained by selectively refining elements from the coarse grid (see Figure 1). The set $\partial\mathcal{P}_H$ of edges in the coarse partition is defined as before. However, the definition of the set $\partial\mathcal{P}_h$ must be generalized to encompass constrained DOFs between the subdomains. Therefore, the set $\partial\mathcal{P}_h$ consists of the edges joining unconstrained vertices lying on the edges of the subdomains. These edges need not coincide with the actual interelement edges as, for instance, is the case in the example shown in Figure 3.

Suppose that the elements contained in the subdomain Ω_K are quasiuniform of size h_K. Associated with each element in the subdomain are five parameters describing the polynomial degree of the approximation on the interior and the four sides of the element. The maximum polynomial degree used on the subdomain Ω_K is denoted by p_K. For technical reasons, we shall demand that the polynomial degree on the boundary of the subdomain does not exceed the polynomial degree anywhere on the interior.

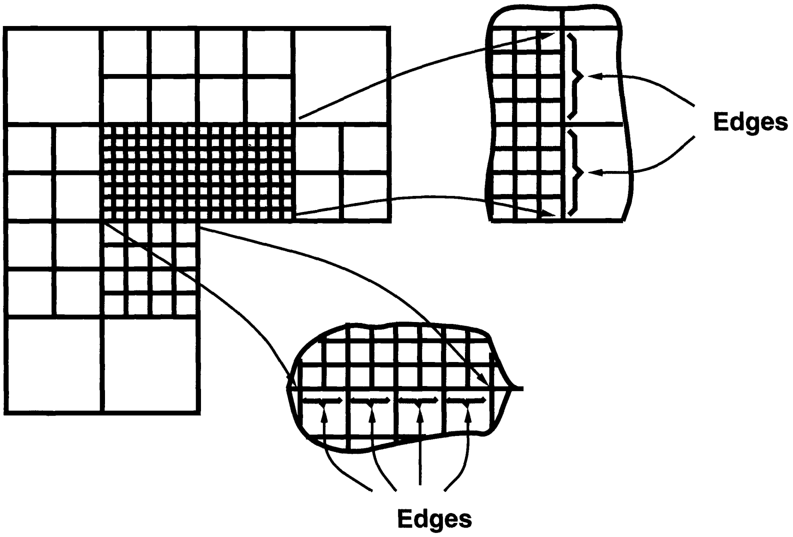


FIG. 3. *Illustration of the edges contained in the set $\partial\mathcal{P}_h$ for a locally refined mesh.*

The analysis of the preconditioner now follows from the previous results obtained for the uniform case as follows. First, the results in Lemma 3.4 are valid for each subdomain separately. However, these results have been proved only under the assumption that the polynomial degree on the boundary does not exceed the polynomial degree on the interior. Hence, Lemmas 3.5 and 3.6 also hold with p replaced by p_K and h replaced by h_K. Arguing as in the proof of Theorem 3.7 then yields the following.

THEOREM 3.9. *Suppose that on each of the refined subdomains Ω_K the mesh is quasi uniform of size h_K and the maximum polynomial degree is p_K. Suppose that the polynomial degree on the subdomain is such that the degree on the boundary does not exceed the polynomial degree on the interior. Then the condition number of the preconditioned system $\kappa(C^{-1}S)$ is bounded by*

$$\kappa(C^{-1}S) \leq K \min\left(\frac{H}{\underline{h}}, 1 + \log^2 \overline{p}\right) \cdot (1 + \log(H\overline{r}))^2 \tag{91}$$

where $\overline{p} = \max_K p_K$, $\underline{h} = \min_K h_K$, and $\overline{r} = \max_K p_K / h_K$. The constant K is independent of the parameters H, $\underline{h}$, and $\overline{p}$ and is independent of the magnitude of the jump discontinuities of the coefficients between neighbouring subdomains.

REFERENCES

[1] M. AINSWORTH, *A Preconditioner Based on Domain Decomposition for h-p Finite Element Approximation on Quasi-Uniform Meshes*, SIAM J. Numer. Anal., 33 (1996), to appear.

[2] I. BABUSKA, A. CRAIG, J. MANDEL, AND J. PITKARANTA, *Efficient preconditioning for the p-version finite element method in two dimensions*, SIAM J. Numer. Anal., 28 (1991), pp. 624–661.

[3] I. BABUSKA AND B. Q. GUO, *The h-p version of the finite element method for domains with curved boundaries*, SIAM J. Numer. Anal., 25 (1988), pp. 837–861.

[4] I. BABUSKA, I. N. KATZ, AND B. SZABO, *The p version of the finite element method*, SIAM J. Numer. Anal., 18 (1981), pp. 515–545.

[5] I. BABUSKA AND M. SURI, *The h-p version of the finite element method with quasi-uniform meshes*, RAIRO Modél. Math. Anal. Numér., 21 (1987), pp. 199–238.

[6] J. H. BRAMBLE, R. E. EWING, J. E. PASCIAK, AND A. H. SCHATZ, *A preconditioning technique for the efficient solution of problems with local grid refinement*, Comput. Methods Appl. Mech. Engrg., 67 (1988), pp. 149–159.

[7] R. K. Coomer and I. G. Graham, *Massively parallel methods for semiconductor device modelling*, Computing, 56 (1996), pp. 1–28.

[8] L. Demkowicz, J. T. Oden, W. Rachowicz, and O. Hardy, *Toward a universal h-p adaptive finite element strategy: Part* 1. *Constrained approximation and data structure*, Comput. Methods Appl. Mech. Engrg., 77 (1989), pp. 113–180.

[9] M. Dryja and O. B. Widlund, *Towards a unified theory of domain decomposition algorithms for elliptic problems*, in Third International Symposium on Domain Decomposition Methods for Partial Differential Equations, T. F. Chan, R. Glowinski, J. Periaux, and O. Widlund, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1989, pp. 3–22.

[10] J. L. Lions and E. Magenes, *Non-Homogeneous Boundary Value Problems and Applications* I, Springer-Verlag, Berlin, New York, 1972.

[11] J. Mandel, *Iterative solvers by substructuring for the p-version finite element method*, Comput. Methods Appl. Mech. Engrg., 80 (1990), pp. 117–128.

[12] J. T. Oden, A. Patra, and Y. S. Feng, *An adaptive h-p strategy*, in Adaptive Multilevel and Hierarchical Computational Strategies, A. K. Noor, ed., ASME Publications, 157 (1992), American Society of Mechanical Engineers, New York, pp. 23–46.

[13] B. F. Smith, *Domain Decomposition Algorithms for the Partial Differential Equations of Linear Elasticity*, Ph.D. thesis, Mathematics Department, New York University, 1991.

[14] J. Xu, *Iterative methods by space decomposition and subspace correction*, SIAM Rev., 34 (1992), pp. 581–614.

SIAM J. SCI. COMPUT.
Vol. 17, No. 6, pp. 1414–1433, November 1996

007

STATISTICAL EQUILIBRIUM COMPUTATIONS OF COHERENT STRUCTURES IN TURBULENT SHEAR LAYERS*

BRUCE TURKINGTON† AND NATHANIEL WHITAKER‡

Abstract. A numerical method is developed to treat the statistical equilibrium model of coherent structures in two-dimensional turbulence. In this model the vorticity, which fluctuates on a microscopic scale, is described macroscopically by a local probability distribution. A coherent vortex is identified with a most probable macrostate, which maximizes entropy subject to the constraints dictated by the complete family of conserved quantities for incompressible, inviscid flow. Attention is focused on the special case corresponding to vortex patches, and a simple, robust, and efficient algorithm is proposed in this case. The form of the iterative algorithm and its convergence properties are derived from the variational structure of the statistical equilibrium problem. Solution branches are computed for the shear layer configuration, and the results are interpreted in terms of the dynamical phenomena of rollup and coalescence.

Key words. vorticity dynamics, maximum entropy states, optimization method

AMS subject classifications. 76C05, 76F99, 82B21, 49M45

1. Introduction.

The emergence of organized or coherent structures in a two-dimensional fluid flow at a high Reynolds number is observed in both physical and computational experiments. In these turbulent fields a macroscopic (coarse-grained) order is discerned in the presence of a microscopic (fine-grained) disorder. Such a coherent structure generally takes the form of a long-lived, large-scale vortical mean flow consisting of either a single, isolated vortex or a regular system of vortices. The vorticity field itself, however, typically exhibits rapid, local fluctuations of the same order of magnitude as the mean vorticity. The extensively studied free shear layer can be chosen as a representative of this general phenomenon. After a complex transient process, a perturbed free shear layer tends to roll up into a sequence of vortices whose structure depends upon the large-scale features of the initial layer but not on the precise nature of the small-scale vorticity fluctuations that result from the intervening dynamical evolution. The final vortex structure then persists as a nearly steady mean flow, changing slowly due to the dissipation dictated by a finite Reynolds number.

In constructing a theoretical explanation of this phenomenon, it is natural to invoke the methods of statistical mechanics. Under the simplifying assumption that the fluid flow is ideal, which is reasonable provided that the Reynolds number is sufficiently high, the governing dynamics are conservative. Consequently, it is possible to adopt a statistical equilibrium model [5, 14]. In such a framework a coherent structure is identified with a macroscopic equilibrium state corresponding to the complete family of conserved quantities associated with the microscopic evolution. This kind of formulation is proposed in the fundamental work of Miller [18] and Miller, Weichman, and Cross [19] and Robert [23] and Robert and Sommeria [24]. These two investigations derive the same statistical equilibrium theory for the two-dimensional flow of an incompressible and inviscid fluid, although their presentations and emphases differ in some respects. The two key ingredients of this theory are (1) a macroscopic description of the vorticity field as a local probability distribution, and (2) a weak form of the conservation of all generalized enstrophies and a mean-field form of the conservation of kinetic

*Received by the editors July 7, 1993; accepted for publication (in revised form) June 16, 1995.

†Department of Mathematics and Statistics, University of Massachusetts, Amherst, MA 01003 (turk@math.umass.edu). The work of this author was partially supported by the National Science Foundation grant DMS-9103976.

‡Department of Mathematics and Statistics, University of Massachusetts, Amherst, MA 01003 (whitaker@math.umass.edu). The work of this author was partially supported by the National Science Foundation grant DMS-8913482.

energy. The equilibrium state furnished by the theory is the most probable macrostate, which has the largest number of microscopic realizations.

The formulation of the Miller–Robert theory rests on a separation of scales for the macroscopic and microscopic descriptions of the vorticity field. In contrast to the theory of homogeneous turbulence of a driven and dissipative fluid, in which energy and enstrophy cascades over a range of intermediate scales are postulated, this theory considers macroscopic variations on the scale of the domain size and microscopic fluctuations on an infinitesimal scale at each point in that domain. This idealization of the turbulent fluctuations, in which correlations are ignored at every finite scale, amounts to a simple, random model of the complex local distortions suffered by the vorticity on increasingly small scales as time increases. Thus, the theory is designed to capture the large-scale features of the vorticity field, and hence the flow, in a relaxed state after a very long evolution at a very high Reynolds number. Analytical and numerical evidence suggests that this model of the local microscopic evolution is in good qualitative agreement with the actual vorticity dynamics. In order to appraise its quantitative validity, however, its predictions must be computed and then compared with direct simulations or experimental observations. Some initial comparisons of this kind are provided by Sommeria, Staquet, and Robert [27] and Robert and Sommeria [25].

Our purpose in this paper is to develop an effective numerical method of solving these statistical equilibrium problems. We base our computations on the constrained maximum entropy formulation of Robert [23] and Robert and Sommeria [24], which seems, to us, to be the natural one from the standpoint of computation. The objective functional is the entropy of an admissible macrostate, which quantifies the multiplicity of the microscopic realizations of that macrostate. The constraint functionals are the (mean-field) energy, the (weak) generalized enstrophies, and whatever other functionals arise from special spatial symmetries. In the present paper we concentrate on the special case in which the initial microstate is assumed to be a vortex patch, meaning that the vorticity takes on the given constant values 0 and λ only.

A relaxation method of producing the maximum entropy state is given by Robert and Sommeria [25]. Their method is based on a macroscopic evolution equation that is designed to conserve all the dynamical constraints and to increase the entropy. A numerical integration over a sufficiently long time interval then provides an approximation to the statistical equilibrium state. Moreover, the evolution equation itself is of interest as a model of subgrid-scale mixing driven by the ideal dynamics.

Our numerical method of solution is an iterative algorithm derived from the optimization structure of the problem at hand. The iterative step is defined by solving a variational subproblem in which the nonlinear equality constraint on energy, which presents the principal difficulty, is replaced by an inequality constraint on its linearization about the previous iterate. We prove that our algorithm converges to a solution from any admissible initialization and that this solution satisfies all the constraints exactly. We obtain an efficient implementation of our algorithm by appealing to the dual form of the iterative subproblem.

In our implemented computations, we display branches of solutions to the statistical equilibrium problem for a periodic free shear layer of constant vorticity. We show how these branches connect the completely disordered state, which is spatially homogeneous, to the deterministic state, which is an exact steady flow. By so doing, we embed the deterministic theory of coherent structures into the statistical theory as a limiting case. Moreover, in the process of computing these branches, we exhibit a symmetry-breaking bifurcation in the statistical equilibrium problem for the free shear layer, which we interpret as an explanation of the rollup phenomenon. In the same context, we apply our method to the coalescence phenomenon for like-signed vortices in a (singly) periodic array.

2. Statistical equilibrium model.

2.1. Vorticity dynamics. The two-dimensional flow of an incompressible, inviscid fluid is described by the Euler equations

$$\frac{\partial v}{\partial t} + v \cdot \nabla v + \nabla p = 0, \quad \nabla \cdot v = 0, \tag{2.1}$$

where $v = (v_1(x,t), v_2(x,t), 0)$ and $p = p(x,t)$ are the velocity and pressure fields, and $\nabla = (\partial/\partial x_1, \partial/\partial x_2, 0)$ with $x = (x_1, x_2)$. In the presence of solid boundaries, these equations hold within a domain $D \subseteq \mathbf{R}^2$, with the boundary condition $n \cdot v = 0$ imposed on ∂D, where n denotes the outward unit normal field. (If, instead, periodic boundary conditions are imposed on either x_1 or x_2 or both, then the domain D is determined by the fundamental period for those conditions.) It is widely recognized that the salient features of these fluid dynamics are more neatly described by the vorticity field $\omega = (0, 0, \omega(x,t))$ than by the primitive fields v and p. Indeed, the governing dynamics reduce to the single equation

$$\frac{\partial \omega}{\partial t} + v \cdot \nabla \omega = 0, \tag{2.2}$$

which expresses the transport of vorticity by the flow. The velocity field is determined from the vorticity field by the elliptic system

$$\nabla \times v = \omega, \quad \nabla \cdot v = 0 \quad \text{in } D. \tag{2.3}$$

This system is easily solved by means of the streamfunction $\psi = \psi(x,t)$, which is defined according to $v = \nabla \times (0, 0, \psi)$. By virtue of the fact that $\omega = -\Delta\psi$, where Δ is the Laplacian operator in $\mathbf{R}^2$, the streamfunction is determined from the vorticity by

$$\psi(x) = G\omega(x) = \int_D g(x, x')\omega(x')dx', \tag{2.4}$$

where G is the Green operator for $-\Delta$ in D with boundary conditions $\psi = 0$ on ∂D (or suitable periodic boundary conditions), and $g(x, x')$ is the corresponding Green function.

The initial value problem for (2.2)–(2.3) is known to be well posed in the classical sense [12]. Nevertheless, the dynamics of vorticity transport tend to produce a very rapid growth in the vorticity gradient $|\nabla\omega(x,t)|$ in active regions of the flow, even though it preserves $\max_{x \in D} \omega(x,t)$ and $\min_{x \in D} \omega(x,t)$ exactly. Such a growth is consistent with the sharpest estimates provided by the existence and uniqueness theory for Euler flow [29]. It is natural, therefore, to consider weak solutions $\omega \in L^\infty(D \times [0,T])$ which are permitted to fluctuate on arbitrarily small scales. While these weak solutions $\omega(x,t)$ induce velocity fields $v(x,t)$ whose regularity properties are sufficient to determine smooth particle trajectories, they are very complex in the sense that their flow maps tend to become increasingly distorted over time at a very rapid rate. This body of qualitative theory, therefore, suggests that rearrangement of vorticity under ideal dynamics generates spatial fluctuations of ω on increasingly small scales.

This qualitative behavior of vortex dynamics, besides being indicated by analytical considerations, is strongly supported by numerical evidence. The many simulations of turbulent flows in two dimensions (Navier–Stokes flows at high Reynolds numbers) display the effects of distortion on the small scales of the vorticity, at least to the extent of the numerical resolution [17, 3, 10]. Similarly, the computations of those weak solutions for which the vorticity takes constant values 0 and λ only, which are referred to as *vortex patches*, show the same effects

[30, 6]. In particular, the free boundary of the vortex patch evolves in a very intricate way, exhibiting the characteristic phenomenon of filamentation on small scales [7].

This theoretical and computational evidence suggests that the deterministic dynamics be viewed as governing the *microscopic state* of the flow. The microstate $\omega(x, t)$ then constitutes the*fine-grained* description of the flow field. Since the information content of this deterministic evolution grows very rapidly over time, due to the complexity of the vorticity fluctuations on increasingly small scales, it does not provide a convenient description of the longtime behavior of the flow. Instead, a *coarse-grained* description of the vorticity is needed whenever an understanding of the long-lived, large-scale vortex structures inherent in the dynamics is desired. Such a description is furnished by a *macroscopic state* that only partially represents the small-scale behavior of the vorticity field. In the formulation of the statistical equilibrium theories proposed in [18, 19] and [23, 24], a macrostate is taken to be a local probability distribution on the values of the vorticity. A coherent structure is then identified with a macrostate that varies on the large scale.

2.2. Microscopic and macroscopic descriptions. We now proceed to define these concepts precisely and to formulate the statistical equilibrium theory in terms of them. For a complete motivation and justification of this theory we refer the reader to the papers cited above. In order to make our presentation accessible, we first explain the theory in the general case of arbitrary microstates and then derive the special case in which the microstates are vortex patches.

Let $I \subseteq \mathbf{R}$ be a finite interval (say the minimal interval) containing the range of the initial vorticity $\omega_0(x)$. Then, $\omega(x, t) \in I$ for all times $t > 0$, by virtue of (2.2). This invariance property allows us to define a *microstate* to be any measurable function $\omega(x)$ taking values in I and a *macrostate* to be any probability measure $p(x, dy)$ on I for almost all $x \in D$. We make a correspondence between these two levels of description by interpreting $p(x, dy)$ as the probability that the fluctuating function ω takes values in dy when sampled at points that are near x. This correspondence is determined by the identity

$$m\{x \in A : \omega(x) \in B\} = \int_A dx \int_B p(x, dy), \tag{2.5}$$

which holds for all measurable subsets $A \subseteq D$ and $B \subseteq I$. It follows that

$$p(x, B) = \lim_{\epsilon \to 0} \frac{m\{x' \in A_\epsilon(x) : \omega(x') \in B\}}{m(A_\epsilon(x))},$$

choosing $A_\epsilon(x)$ to be a neighborhood of x for which $\operatorname{diam} A_\epsilon(x) \leq \epsilon$. The interpretation of p as a volume fraction is clear from the latter expression. The expectation of the distribution p defines the mean vorticity

$$\bar{\omega}(x) = \int_I y \, p(x, dy). \tag{2.6}$$

The above correspondence between macrostates and microstates can be elucidated by considering the longtime evolution of vorticity. Over any finite interval $0 \leq t \leq T$, we can identify the unit point measure $p(x, dy, t) = \delta_{\omega(x,t)}(dy)$ concentrated at $y = \omega(x, t)$ with the evolving microstate $\omega(x, t)$. But we cannot maintain this identification in the limit as $t \to +\infty$ because of the hypothesis that the vorticity develops finite oscillations on arbitrarily small scales. If a coherent structure emerges in this evolution, then we identify it with the *weak limit* $\omega^*(x)$ of the microstate, and we describe it macroscopically by the corresponding *weak limit* $p^*(x, dy)$. Under weak convergence, the limit $p^*(x, dy)$ is not expected to be concentrated at a single point ($y = \omega^*(x)$) but rather is treated as a general probability distribution with mean $\omega^*(x)$. In other words, the x-parametrized measure $p^*(x, dy)$ encodes the local oscillations of the limiting vorticity field in a way that is compatible with weak convergence. As such, it is a Young measure [8].

2.3. Maximum entropy principle. In the theory of statistical equilibrium, we seek the most probable (most random) macrostate. Such a macrostate is characterized by maximizing an entropy functional, which quantifies the loss of information incurred by going from a microscopic to a macroscopic description. In the absence of any information that constrains the spatial variations of the admissible macrostates, the assumed ergodicity of the vorticity dynamics implies that the most probable macrostate is spatially homogeneous. This x-independent macrostate, which can be viewed as a complete mixing of the initial microstate ω_0, is simply given by

$$p_0(B) = \frac{m\{x \in D : \omega_0(x) \in B\}}{m(D)} \tag{2.7}$$

for measurable sets $B \subseteq I$. However, p_0 is not admissible, in general, because it violates the constraint dictated by the conservation of energy. Therefore, we define the (Kullback–Leibler) *entropy* of any macrostate $p(x, dy)$ relative to $p_0(dy)$ by

$$S(p) = -\int_D dx \int_I \log \frac{dp}{dp_0}(x, y)\, p(x, dy), \tag{2.8}$$

where dp/dp_0 is the density (or Radon–Nikodým derivative) of p with respect to p_0. The information functional $-S(p)$ can be interpreted as a statistical distance between p and p_0. Consequently, we formulate the statistical equilibrium theory by taking the most probable macrostate p to be the maximizer of S subject to *all* of the constraints implied by the vorticity transport equation. This equilibrium macrostate, unlike p_0, exhibits the spatial variations characteristic of a coherent structure.

The constraints imposed in the maximum entropy principle are dictated by the conserved quantities for ideal fluid dynamics. Expressed in terms of the microstate, these are the *energy* and the *generalized enstrophies*, respectively,

$$E(\omega) = \frac{1}{2}\int_D \omega G \omega\, dx, \tag{2.9}$$

$$F(\omega; f) = \int_D f(\omega)\, dx, \tag{2.10}$$

where f is any continuous function on I. Apart from those additional conserved quantities which are associated with special symmetries of the domain (and the boundary conditions), these are the only conserved quantities for ideal fluid flow in two dimensions. In order to impose these constraints in the statistical equilibrium problem, which is formulated in terms of the macrostate, it is necessary to express them as functionals of p. In these expressions the continuity with respect to weak convergence (in either ω or p) must be ensured, since this mode of convergence is expected as $t \to +\infty$. For this reason, the correct forms for these expressions are given by

$$E(p) = \frac{1}{2}\int_D \bar{\omega} G \bar{\omega}\, dx, \tag{2.11}$$

$$F(p; f) = \int_D dx \int_I f(y)\, p(x, dy). \tag{2.12}$$

The energy expression (2.11) is justified by the the compactness of the operator $G : L^2(D) \to L^2(D)$, which implies that the functional E is weakly continuous. Thus, the energy of a macrostate p is determined by the mean streamfunction $\bar{\psi} = G\bar{\omega}$. This fact gives the model the character of a mean-field theory [18, 19]. On the other hand, the generalized enstrophy functionals, even in the classical case for which $f = \frac{1}{2}\omega^2$, are not continuous with respect to weak convergence of ω, except when f is linear, which gives the circulation functional

$$(2.13) \qquad C(p) = \int_D \bar{\omega}\, dx\,.$$

In general, the appropriate relaxed form (2.12) must be taken as the definition of generalized enstrophy of the macrostate p [23, 24]. It is a central fact in the theory of Young measures that the functional (2.12) furnishes the limit of the functionals (2.10) under weak convergence [8].

With the objective functional S and the constraint functionals E and F in hand, we can state the variational principle for statistical equilibrium macrostates p:

$$(2.14) \qquad S(p) \to \max \quad \text{subject to} \quad E(p) = E_0\,, \quad F(p; f) = F_0(f)\,,$$

where the constraint values E_0 and $F_0(f)$ are derived from a given initial vorticity field ω_0. The f-parametrized family of constraints on generalized enstrophy can be expressed equivalently as the condition

$$(2.15) \qquad m(D)^{-1} \int_D p(x, dy)\, dx \;=\; p_0(dy)\,,$$

meaning that the x-average of p over D is the given distribution p_0. Of course, the constraint that $p(x, dy)$ be a probability measure for almost every $x \in D$ is also enforced. By standard arguments in functional analysis, the existence of a maximizing macrostate can be established. The equilibrium equations satisfied by such a solution are

$$(2.16) \qquad p(x, dy) = \frac{\exp(-\alpha(y) - \beta y G\bar{\omega}(x)) p_0(dy)}{\int_I \exp(-\alpha(y) - \beta y G\bar{\omega}(x)) p_0(dy)}$$

for some multipliers $\alpha(y)$ and β, which are determined by p. This result can be derived formally by a standard calculation, which is omitted. The mean flow for this equilibrium satisfies the semilinear elliptic equation

$$(2.17) \qquad -\Delta\bar{\psi} = \Phi'(\bar{\psi}) \quad \text{with} \quad \Phi(s) = -\beta^{-1} \log \int_I \exp(-\alpha(y) - \beta y s) p_0(dy).$$

These equations are supplemented by the constraints on E and F, which determine the multipliers β and $\alpha(y)$.

2.4. Statistical equilibria for vortex patches. Rather than further pursue the general formulation described above, we now specialize our development to the simplest case when ω_0 takes the values 0 and λ only, where λ is a given positive constant. For such vortex patches the microscopic evolution is described by a weak solution $\omega \in L^\infty(D \times [0, T])$ that also takes the values 0 and λ only. We therefore write $\omega(x, t) = \lambda 1_{\Omega(t)}$ where $\Omega(t)$ is an open subset of D, and $1_\Omega = 1$ in Ω, $1_\Omega = 0$ in $D\backslash\Omega$; so, $\omega_0 = \lambda 1_{\Omega(0)}$. The conservation of (total) circulation C alone replaces the family of generalized enstrophy integrals, which is degenerate in this case. Thus, the patches $\Omega(t)$ evolve in such way as to conserve their area $C_0/\lambda = m(\Omega(0))$. The macroscopic description simplifies in this case to $p(x, dy) = (1 - \rho(x))\delta_0(dy) + \rho(x)\delta_\lambda(dy)$, in which $\rho(x)$ is the probability that $\omega(x') = \lambda$ for x' near x, or, equivalently, the volume fraction occupied by Ω near x. The mean vorticity is simply $\bar{\omega}(x) = \lambda\rho(x)$.

The statistical equilibrium problem for vortex patches, which is a reduction of the general problem (2.14), can be stated in the form

$$S(\rho) := -\int_D [(1-\rho)\log(1-\rho) + \rho\log\rho]dx \quad \to \quad \max \tag{2.18}$$
$$\text{subject to} \quad C(\rho) = C_0, \quad E(\rho) = E_0.$$

This entropy functional differs from that defined in (2.8) by a constant depending only on C_0 and ρ_0, as can be checked by noting that the density dp/dp_0 equals $(1-\rho)/(1-\rho_0)$ at $y = 0$, and ρ/ρ_0 at $y = \lambda$, if $p_0(dy) = (1-\rho_0)\delta_0(dy) + \rho_0\delta_\lambda(dy)$. For the sake of simplicity, therefore, we choose the classical (Shannon) entropy as the objective functional. The constraint functionals E and C depend upon $\bar{\omega}$ as in their definitions (2.11) and (2.13), respectively.

An equilibrium solution to (2.18) satisfies the equation

$$\rho(x) = \frac{\exp(-\alpha\lambda - \beta\lambda^2(G\rho)(x))}{1 + \exp(-\alpha\lambda - \beta\lambda^2(G\rho)(x))} \tag{2.19}$$

for some real multipliers α and β. From this equation it follows that the mean flow in equilibrium satisfies

$$-\Delta\bar{\psi} = \Phi'(\bar{\psi}) \quad \text{with} \quad \Phi(s) = -\beta^{-1}\log(1 + \exp(-\alpha\lambda - \beta\lambda s)). \tag{2.20}$$

This semilinear elliptic equation is of eigenvalue type in which the multipliers α and β act as the eigenvalue parameters. In the typical situation of interest the coherent structure is a coalesced vortex and, necessarily, the inverse temperature β is negative. The statistical equilibrium problem can then have multiple solutions and bifurcating solution branches. This makes the direct solution of the mean-field equation (2.20) difficult.

The maximum entropy principle as stated above admits various extensions in the presence of additional constraints. These constraints correspond to special symmetries in the domain geometry and flow configuration. In this paper we are concerned with the shear layer problem for which the coherent structure consists of an infinite system of vortices repeated periodically in the x_1 direction and wall bounded in the x_2 direction. Consequently, we take $D = \{x \in \mathbf{R}^2 : a_1 < x_1 < b_1,\ a_2 < x_2 < b_2\}$, and for boundary conditions we impose periodicity on the sides $x_1 = a_1$ and $x_1 = b_1$ and $\bar{\psi} = 0$ on the sides $x_2 = a_2$ and $x_2 = b_2$. The x_1-translational symmetry of this configuration yields the conserved quantity

$$M = \int_D x_2\omega\, dx, \tag{2.21}$$

which is the x_1 component of linear impulse [1]. In the variational principle for statistical equilibrium, this conserved quantity introduces the additional constraint $M(\rho) = M_0$, where $M(\rho)$ is defined as in (2.21) with $\bar{\omega}$ replacing ω. In the equilibrium equations (2.19) and (2.20), the streamfunction $\bar{\psi}$ is then replaced by $\bar{\psi} + (\gamma/\beta)x_2$, which includes an x_1-translational flow with velocity $-\gamma/\beta$, where γ is the multiplier for the linear impulse constraint.

3. Deterministic equilibrium theory.

3.1. Feasible constraint values. The admissible class for the maximum entropy principle is void for constraint values outside a certain feasible range. The limits of this range are defined by the deterministic equilibrium theory of coherent structures, which itself has a variational formulation. As the constraint values approach the limits of the feasible range, the statistical equilibrium solutions tend to their deterministic analogues. In the case of vortex patches, the constraint values C_0 and E_0 in the maximum entropy principle (2.18) cannot be prescribed arbitrarily. Rather, for given values of C_0 and λ, there is a finite interval $[E_{\min}, E_{\max}]$ of feasible values of E_0. The upper and lower bounds of this range of feasible energy values are defined by deterministic coherent structures which have the property that $\rho(x)$ is either 0 or 1 for almost all $x \in D$. In order to explain this useful connection between the statistical and deterministic theories of coherent structures, we summarize below the variational theory of steady vortex flows which characterizes these limits exactly.

The upper bound for E_0 in the statistical equilibrium problem (2.18) is given by the variational principle developed in [28]. In this principle, a steady vortex patch $\omega \in L^\infty(D)$ is a solution of the constrained maximization problem

$$E(\omega) \to \max \quad \text{subject to } C(\omega) = C_0, \; 0 \leq \omega \leq \lambda. \tag{3.1}$$

As is shown in [28], a maximizer $\omega = \omega_{\max}$ exists and has the form

$$\omega = \lambda 1_\Omega, \quad \text{where} \quad \Omega = \{x \in D : G\omega(x) > \mu\} \tag{3.2}$$

for a multiplier μ determined by the circulation constraint. (The admissible class of vorticities in (3.1) admits those ω which take values in the interval $[0, \lambda]$, while the maximizer $\omega_{\max}$ equals either 0 or λ almost everywhere in D.) We, therefore, see that the energy constraint in the maximum entropy principle must be prescribed with the restriction that $E_0 \leq E_{\max} := E(\omega_{\max})$.

The limiting behavior of the statistical equilibrium solutions $\bar{\omega}$ as $E_0 \to E_{\max}$ can be determined under the condition that $\omega_{\max}$ is an isolated and nondegenerate maximizer in (3.1), meaning that $\int_D |\omega_{\max} - \omega| \, dx$ tends to zero as $E(\omega)$ approaches $E_{\max}$. Under such a condition the deterministic steady vortex patch is dynamically stable with the Lyapunov functional $E_{\max} - E(\omega)$. Since $\omega_{\max}$ and $\bar{\omega}$ are uniformly bounded in D, this stability condition expressed in the (circulation) L^1-norm implies that $\bar{\omega} \to \omega_{\max}$ in the (enstrophy) L^2-norm as $E_0 \to E_{\max}$. The limiting behavior of the multipliers α and β is inferred from the following expression for the pointwise difference:

$$\omega_{\max} - \bar{\omega} = \begin{cases} \lambda[1 + \exp(-\alpha\lambda - \beta\lambda\bar{\psi})]^{-1} & \text{in } \{\psi_{\max} > \mu\}, \\ \lambda[1 + \exp(\alpha\lambda + \beta\lambda\bar{\psi})]^{-1} & \text{in } \{\psi_{\max} < \mu\}, \end{cases}$$

which results from combining the equilibrium equations (2.19) and (3.2). In the limit, therefore, $-\alpha - \beta\bar{\psi} \to \infty$ in $\{\psi_{\max} > \mu\}$, while $\alpha + \beta\bar{\psi} \to \infty$ in $\{\psi_{\max} < \mu\}$. This, together with the fact that $\bar{\psi} \to \psi_{\max}$ uniformly on $\bar{D}$, which holds by virtue of the regularity properties of the operator G, shows that $\beta \to -\infty$ and $-\alpha/\beta \to \mu$ as $E_0 \to E_{\max}$. This behavior of the the multipliers is consistent with the fact that the admissible class in (2.18) shrinks to the singleton $\{1_{\{\psi_{\max} > \mu\}}\}$. Thus, the statistical equilibrium problem (2.18) degenerates into the deterministic equilibrium problem (3.1) in the limit of maximum feasible energy. This behavior is clearly displayed by the numerical results given in §5.

The asymptotic behavior of the $E_{\max}$ and $\omega_{\max}$ as $\lambda \to \infty$ for fixed C_0 is also known [28]. In this limit, the statistical equilibrium solutions tend to the solutions of the classical theory of a dilute point-vortex gas [21, 20, 16, 13, 4]. The equilibrium equation then reduces to the mean-field equation

$$\omega = \exp(-\tilde{\alpha} - \tilde{\beta} G\omega) \tag{3.3}$$

for the vortex density ω. The deterministic solution $\omega_{\max}$, on the other hand, tends to a point vortex with circulation C_0 located at an equilibrium point of the Kirchhoff–Routh Hamiltonian for the domain D. The corresponding energy $E_{\max}$ diverges to infinity logarithmically, and hence arbitrarily large values of E_0 are feasible in the classical theory. As E_0 increases to infinity these classical statistical equilibria condense to a point vortex, which constitutes their deterministic limit.

The lower bound for E_0 in (2.18) is given by the deterministic equilibrium problem of minimum energy analogous to (3.1). Unlike the coalesced vortices obtained at maximum energy, these minimum energy solutions are steady flows for which the vorticity is dispersed to the domain boundary. The energy of such a solution $\omega_{\min}$ provides the complementary restriction that $E_0 \geq E_{\min} := E(\omega_{\min})$. In the limit of minimum feasible energy, the statistical theory again degenerates into the deterministic theory.

Intermediate between $E_{\min}$ and $E_{\max}$ lies the energy value $E_{\text{hom}} = E(\rho_{\text{hom}})$ of the most probable macrostate ρ_{hom} over all possible energy values. This homogeneous macrostate is simply the constant $\rho_{\text{hom}} = C_0/\lambda m(D)$, which corresponds to $\beta = 0$. In simple geometries, like those of our shear layer computations, the statistical equilibria with $E_{\text{hom}} < E_0 < E_{\max}$ have negative temperatures ($\beta < 0$), while those with $E_{\min} < E_0 < E_{\text{hom}}$ have positive temperatures ($\beta > 0$). In complex geometries, however, this may not always hold.

3.2. Iterative algorithm. The deterministic equilibrium problem (3.1) can be solved numerically using the method given in [9]. Since that method can be viewed as a prototype for the iterative algorithm that we propose in the present paper to solve the statistical equilibrium problem (2.18), we briefly restate it here.

The algorithm generates the iterate ω^{k+1} ($k = 0, 1, \ldots$) by solving the optimization problem that results from linearizing the energy functional about the previous iterate ω^k. Thus, $\omega = \omega^{k+1}$ is taken to be the solution of the convex subproblem

$$\int_D \omega G\omega^k dx \;\to\; \max \quad \text{over } C(\omega) = C_0, \quad 0 \leq \omega \leq \lambda. \tag{3.4}$$

The variational equations for this subproblem are simply

$$\omega^{k+1} = \lambda 1_{\Omega^{k+1}} \quad \text{where} \quad \Omega^{k+1} = \{x \in D : G\omega^k(x) > \mu^{k+1}\} \tag{3.5}$$

for some multiplier μ^{k+1}. This multiplier is found by imposing the circulation constraint on the expression (3.5), which yields a single real equation that is easily solved. Formally, the dual subproblem to the primal subproblem (3.4),

$$C_0\mu + \int_D \lambda(G\omega^k - \mu)_+ dx \;\to\; \min \quad \text{over} \quad \mu \in \mathbf{R},$$

defines the iterative multiplier μ^{k+1}.

This iterative algorithm is designed to be globally convergent, meaning that it converges from *any* admissible initialization ω^0. This property can be inferred from the inequality

$$E(\omega^{k+1}) - E(\omega^k) \geq E(\omega^{k+1} - \omega^k),$$

which holds for all k. A general convergence theory is presented in [9]. The algorithm has a linear rate of convergence, which is observed to be quite rapid in implemented computations.

4. Numerical method.

4.1. Primal form of the algorithm. We now proceed to give a method of computing solutions of the maximum entropy problem posed in §2. This method is an iterative algorithm based on the special structure of the constrained optimization problem that it solves. In this sense, it fits into the same conceptual framework as the algorithm described in §3 which solves the related maximum energy problem.

The iterative step that generates ρ^{k+1} from ρ^k $(k = 0, 1, \ldots)$ is defined by solving the optimization subproblem

$$S(\rho) \to \max \quad \text{subject to} \quad C(\rho) = C_0, \quad E(\rho^k) + \langle E'(\rho^k), \rho - \rho^k \rangle \geq E_0. \tag{4.1}$$

Given ρ^k, the solution $\rho = \rho^{k+1}$ to this subproblem exists and is unique, since the objective functional is strictly concave and the admissible class is convex. Consequently, we obtain a well-defined iterative sequence ρ^k $(k = 0, 1, \ldots)$ for every admissible initialization ρ^0 we choose. We require only that ρ^0 satisfy a relaxation of the constraints for (2.18): $C(\rho^0) = C_0$ and $E(\rho^0) \geq E_0$. In this abstract formulation, the functional derivative (denoted by a prime) of any differentiable functional $F(\rho)$ is defined by the condition that

$$F(\rho + \delta\rho) = F(\rho) + \langle F'(\rho), \delta\rho \rangle + o(\| \delta\rho \|)$$

as $\| \delta\rho \| \to 0$, where the symbols $\langle \cdot, \cdot \rangle$ and $\| \cdot \|$ denote the $L^2(D)$ pairing and norm, respectively.

The solution ρ^{k+1} satisfies

$$S'(\rho^{k+1}) = \alpha^{k+1} C'(\rho^k) + \beta^{k+1} E'(\rho^k), \tag{4.2}$$

$$\beta^{k+1} \leq 0, \tag{4.3}$$

$$\beta^{k+1}[E(\rho^k) + \langle E'(\rho^k), \rho^{k+1} - \rho^k \rangle - E_0] = 0 \tag{4.4}$$

for some multipliers $\alpha^{k+1}, \beta^{k+1} \in \mathbf{R}$. These are the Kuhn–Tucker conditions for (4.1), which provide the natural extension of the Lagrange multiplier rule to an inequality constrained problem [26, 11]. The complementarity conditions (4.4) imply that the linearized energy constraint holds as an equality whenever $\beta^{k+1} \neq 0$. The solution triple $(\rho^{k+1}, \alpha^{k+1}, \beta^{k+1})$ is completely determined by these conditions combined with the constraints.

4.2. Convergence properties. Before expressing the iterative algorithm in the concrete form needed for numerical implementation, we first show how its convergence properties are derived from its abstract structure. These special properties justify the algorithm and explain how it makes use of the concavity/convexity properties of the objective/constraint functionals in the variational problem it solves. For this purpose we need the the following second-order expansions of S and E:

$$S(\rho + \delta\rho) \leq S(\rho) + \langle S'(\rho), \delta\rho \rangle - 2 \| \delta\rho \|^2,$$
$$E(\rho + \delta\rho) = E(\rho) + \langle E'(\rho), \delta\rho \rangle + E(\delta\rho).$$

By applying these facts along with the defining equation (4.2), we make the calculation

$$\begin{aligned} S(\rho^{k+1}) - S(\rho^k) - 2 \| \rho^{k+1} - \rho^k \|^2 &\geq \langle S'(\rho^{k+1}), \rho^{k+1} - \rho^k \rangle \\ &= \beta^{k+1} \langle E'(\rho^k), \rho^{k+1} - \rho^k \rangle \\ &= \beta^{k+1} [E_0 - E(\rho^k)] \end{aligned}$$

in which we note that the term involving α^{k+1} vanishes because C is a linear functional. By virtue of the convexity of E, we have

$$E(\rho^k) \geq E(\rho^{k-1}) + \langle E'(\rho^{k-1}), \rho^k - \rho^{k-1} \rangle \geq E_0.$$

Thus, we obtain the basic inequality of the convergence theory:

$$S(\rho^{k+1}) - S(\rho^k) \geq 2 \parallel \rho^{k+1} - \rho^k \parallel^2 + |\beta^{k+1}|[E(\rho^k) - E_0] \tag{4.5}$$

which holds for all k and for arbitrary admissible ρ^0.

We infer immediately that the entropy $S(\rho^k)$ increases along the iterative sequence, and hence that it converges to a finite limit S^* as $k \to \infty$. From this monotonicity property we conclude that $\rho^{k+1} - \rho^k \to 0$ in L^2. This conclusion shows that if the iterative sequence ρ^k converges to a limit ρ^* and the multiplier sequence β^k converges to a limit $\beta^* < 0$, then ρ^* is a solution of the maximum entropy problem with equality constraints on both circulation and energy. Indeed, if we assume that $\rho^k \to \rho^*$ in L^2 as $k \to \infty$, then it is straightforward to demonstrate that the corresponding multipliers converge: $\alpha^k \to \alpha^*$ and $\beta^k \to \beta^*$. We then deduce that the limit $(\rho^*, \alpha^*, \beta^*)$ solves the equation for statistical equilibrium

$$S'(\rho^*) = \alpha^* C'(\rho^*) + \beta^* E'(\rho^*),$$

using the fact that ρ^k and ρ^{k+1} both converge to ρ^*. Given that $\beta^* < 0$, we retrieve the equality constraints on ρ^*, since $C(\rho^k) = C_0$ and $E(\rho^k) \to E_0$.

From the inequality (4.5) alone, however, we cannot conclude that the iterative sequence ρ^k converges since we lack a rate of convergence on the difference between successive iterates. Moreover, the possible nonuniqueness of solutions ρ^* to the statistical equilibrium problem suggests that this convergence may not always hold. (The equality constraint on energy in the maximum entropy problem makes it a nonconvex optimization problem.) For these reasons, we generalize the notion of convergence to one that holds even when the statistical equilibrium problem does not have a unique (or isolated) solution. Simply put, the generalized convergence theorem states that the minimum distance from ρ^k to the *set* of solutions (critical points) ρ^* of (2.18) tends to zero as $k \to \infty$. We now give an outline of the proof of this theorem based on the analysis developed in [9] for the deterministic problem.

The core of this proof is to show that any subsequence of the iteratives has a further subsequence that converges in the energy norm to a solution of a certain relaxed version of the maximum entropy problem. Let the G-norm, which is equivalent to the energy norm, be defined by $\parallel \rho \parallel_G^2 = \int_D \rho \, G\rho \, dx$. Since the sequence ρ^k is bounded in L^2, any sequence ρ^{k_j} of iterates has a subsequence $\rho^{k'_j}$ that converges weakly to some $\rho^* \in L^2$. By virtue of the compactness of the operator G, this subsequence also converges in the G-norm. Moreover, $\rho^{k'_j+1}$ converges to the same limit ρ^* weakly in L^2 and in the G-norm. To prove that ρ^* is a solution to the statistical equilibrium equation (a critical point of the maximum entropy problem), it suffices to show that ρ^* solves the variational problem

$$S(\rho) \to \max \quad \text{subject to} \quad C(\rho) = C_0, \quad E(\rho^*) + \langle E'(\rho^*), \rho - \rho^* \rangle \geq E_0. \tag{4.6}$$

In other words, it is required to show that $S(\rho^*) \geq S(\rho)$ for any ρ that is admissible in (4.6) in which the energy constraint is linearized about ρ^* itself. For any such ρ, let $\eta \in L^\infty(D)$ be chosen such that its support is contained in $\{x \in D : \delta \leq \rho(x) \leq 1 - \delta\}$ for some positive δ, and it satisfies $C(\eta) = 0$, $\langle E'(\rho^*), \eta \rangle = 1$. (This construction is possible unless $S(\rho) = 0$, which is a trivial case.) We claim that the perturbation $\rho + \epsilon\eta$, for any small positive ϵ, is admissible in the iterative subproblem (4.1) whenever $k = k'_j$ is large enough depending on

ϵ. To verify this claim we evaluate the constraints in (4.1) on $\rho + \epsilon\eta$, using the constraints in (4.6) on ρ; thus, we find that

$$\begin{aligned} C(\rho + \epsilon\eta) &= C_0, \\ E(\rho^k) + \langle E'(\rho^k), \rho + \epsilon\eta - \rho^k\rangle &\geq E(\rho^*) + \langle E'(\rho^*), \rho - \rho^*\rangle - \epsilon_k + \epsilon \\ &\geq E_0 - \epsilon_k + \epsilon \end{aligned}$$

for a sequence of positive constants ϵ_k which, by the continuity of the energy functional, tend to zero along the subsequence. Therefore, $S(\rho+\epsilon\eta) \leq S(\rho^{k+1})$ for large $k = k'_j$, and hence in the limit, $S(\rho + \epsilon\eta) \leq S(\rho^*)$, by the weak upper semicontinuity of the entropy functional. Upon taking ϵ to zero, we obtain the desired result that ρ^* solves (4.6), which means that ρ^* satisfies the statistical equilibrium equation and the relaxed constraints: $C(\rho^*) = C_0, \quad E(\rho^*) \geq E_0$.

Let Σ^* be the *set of critical points* ρ^* of the maximum entropy problem (2.18) with the energy constraint relaxed to an inequality. From the above analysis, we obtain the generalized convergence result in terms of the G-norm distance from the iterate ρ^k to the set Σ^*:

$$\text{dist}_G(\rho^k, \Sigma^*) \ := \ \inf_{\rho^* \in \Sigma^*} \parallel \rho^k - \rho^* \parallel_G \ \rightarrow \ 0 \qquad \text{as } k \rightarrow \infty\,.$$

This convergence statement holds without any restrictions on the constraint values C_0 and E_0 or on the initialization ρ^0. In this sense, the iterative algorithm defined by (4.1) is globally convergent. The proof of the statement is immediate, since otherwise $\text{dist}_G(\rho^k, \Sigma^*) \geq \delta > 0$ for a sequence $k = k_j$, which contradicts the convergence of a subsequence $\rho^{k'_j}$ to a critical point $\rho^* \in \Sigma^*$.

It is possible to strengthen this convergence result in several ways. First, by applying a standard bootstrap argument to the statistical equilibrium equation, the G-norm can be replaced by the L^2-norm. Second, for any given initialization ρ^0, the set of limit points of ρ^k can be shown to consist of either a single solution (the sequence limit) or infinitely many solutions, none of which is isolated. Third, the convergence of the (entire) iterative sequence can be ensured by requiring that the initialization be close enough to an isolated local maximizer ρ^* for (2.18). We omit the details of these further results.

The energy constraint, which is relaxed to the inequality $E(\rho^*) \geq E_0$ in the convergence theory, is necessarily an equality whenever $\beta^* < 0$, by virtue of the complementarity conditions. Thus, it is evident that our algorithm is designed to compute solutions having negative temperature. This restriction is completely natural, however, since these solutions are the statistical equilibria of physical interest as coherent structures. Indeed, as in the theory of a point-vortex gas, the coalescence of vorticity occurs only in the negative temperature regime.

In implemented computations, our algorithm exhibits a linear rate of convergence to an isolated solution. This behavior is expected on analytical grounds whenever ρ^* is a nondegenerate local maximizer, which is the generic case. The generalized convergence behavior stated in the global convergence theorem is, therefore, rarely observed. Moreover, we find that the algorithm converges even in the regime where $\beta^* \geq 0$, which is beyond the scope of the convergence theory. As a consequence, actual implementations of the algorithm can dispense with the inequality constraint on linearized energy in favor of the simpler equality constraint.

4.3. Dual form of the algorithm.

We now give the concrete form of the iterative algorithm, which yields its numerical implementation. Equation (4.2), which defines the iterate ρ^{k+1} in terms of the multipliers α^{k+1} and β^{k+1}, can be written in the form

$$\rho^{k+1} \ = \ (s^*)'(\alpha^{k+1}\lambda \ + \ \beta^{k+1}\lambda\bar{\psi}^k) \tag{4.7}$$

where $\bar{\psi}^k = G\bar{\omega}^k, \ \bar{\omega}^k = \lambda\rho^k$. This expression employs the Legendre–Fenchel transform

$$s^*(\sigma) \ := \ \sup_{\rho}\{\rho\sigma - s(\rho)\} \ = \ -\log(1 + \exp(-\sigma))$$

of the entropy density $s(\rho) := -[\rho \log \rho + (1-\rho)\log(1-\rho)]$ in (2.18). The derivation of (4.7) follows directly from the standard properties of conjugate functions:

$$\sigma = s'(\rho) \quad \text{and} \quad \rho = (s^*)'(\sigma),$$
$$(s^*)''(\sigma) = 1/s''(\rho) = -\rho(1-\rho) < 0.$$

In view of the explicit formula (4.7), the iterative step is reduced to finding the iterative multipliers α^{k+1} and β^{k+1}. These multipliers, in turn, solve the dual to the primal subproblem (4.1), which is

$$\alpha C_0 + \beta(E_0 + E^k) - \int_D s^*(\alpha\lambda + \beta\lambda\bar{\psi}^k)dx \to \min \qquad \text{over } \alpha \in \mathbf{R},\ \beta \leq 0, \tag{4.8}$$

where $E^k := \langle E'(\rho^k), \rho^k\rangle - E(\rho^k)$ is identical with $E(\rho^k)$. This dual subproblem is easily solved numerically, at least in principle, since it involves a convex objective function on $\mathbf{R}^2$ with one simple bound constraint (which can be ignored in practice). The unique solution $(\alpha^{k+1}, \beta^{k+1})$ to (4.8) determines the unique solution ρ^{k+1} to (4.1) by means of the formula (4.7).

The equivalence of the dual subproblem (4.8) to the primal subproblem (4.1) is a standard fact from optimization theory [26, 11]. The dual subproblem is derived by substituting the expression (4.7) for ρ^{k+1} into the constraints for the primal subproblem. The variational conditions satisfied by a minimizer $(\alpha^{k+1}, \beta^{k+1})$ for (4.8) are then shown to coincide with the Kuhn–Tucker conditions for (4.1).

In the case of the free shear layer the maximum entropy principle is supplemented by an additional constraint associated with the conservation of the x_1-component of linear impulse. The above algorithm is easily adapted to this case by augmenting the circulation constraint into a pair of linear constraints on C and M and the corresponding multiplier into a multiplier vector (α, γ). The abstract construction of the iterative algorithm then generalizes in an obvious manner, and the general convergence results also continue to hold. The explicit form of the iterative step is then given by

$$\rho^{k+1} = (s^*)'(\alpha^{k+1}\lambda + \beta^{k+1}\lambda\bar{\psi}^k + \gamma^{k+1}\lambda x_2),$$

where the multipliers are determined by solving the dual subproblem

$$\alpha C_0 + \beta(E_0 + E^k) + \gamma M_0 - \int_D s^*(\alpha\lambda + \beta\lambda\bar{\psi}^k + \gamma\lambda x_2)dx \to \min$$
$$\text{over } \alpha, \gamma \in \mathbf{R},\ \beta \leq 0.$$

4.4. Numerical implementation. In the shear layer computations discussed in what follows, the numerical problem is discretized on a regular grid on the rectangular domain D. The streamfunction $\bar{\psi} = \lambda G \rho$ is found by utilizing a Poisson solver with boundary conditions that are periodic in x_1 and homogeneous Dirichlet in x_2. The dual subproblem, which presents the main computational burden, is treated by simply ignoring the inequality constraint on β so that it becomes the system of nonlinear equations

$$\left(\frac{\partial}{\partial\alpha}, \frac{\partial}{\partial\beta}, \frac{\partial}{\partial\gamma}\right) \int_D s^*(\alpha\lambda + \beta\lambda\bar{\psi}^k(x) + \gamma\lambda x_2)\,dx = (C_0, E_0 + E^k, M_0).$$

This system in $(\alpha, \beta, \gamma) \in \mathbf{R}^3$ is solved by a damped Newton method with a damping factor 2^{-m}, where m is taken to be the smallest nonnegative integer for which the residual is decreased in the Euclidean norm. (Without the damping, the Newton method can produce overshoots in the arguments of the exponential functions participating in the dual subproblem, and this can lead to divergence.) The damped Newton iterations are terminated when a residual of less than 10^{-9} is obtained. The double integrals over D that define the component equations of the system are evaluated by a standard numerical integration. This direct approach to solving the dual subproblem is fast and accurate. Moreover, if the multiplier $\beta^* = \lim \beta^k$ is strictly negative, and the initialization ρ^0 is close enough to the solution ρ^*, then it is justified theoretically since necessarily the iterative multipliers β^k are strictly negative and the linearized constraints on energy hold as equalities. In general, however, the relaxation of the constraint $\beta \leq 0$ violates the requirements of our convergence theory. Thus, the justification for this approach rests mainly on experience, which shows that over a wide range of conditions, including those in which β^* itself is positive, the iterative algorithm in this relaxed form is convergent.

The rate of convergence of the iterative algorithm is observed to be quite rapid. The stopping criterion adopted for the iterations in k is of the form

$$\frac{\| \rho^{k+1} - \rho^k \|}{\| \rho^k \|}, \frac{|S^{k+1} - S^k|}{|S^k|}, \frac{|E^{k+1} - E^0|}{|E^0|} \leq 5 \times 10^{-3}.$$

While the number of iterations required to satisfy this criterion varies with the conditions of the problem, typically about 10 iterations suffice. No significant changes are observed as a result of decreasing these or the other tolerances set on the algorithm. Similarly, the spatial resolution achieved by taking 50×50 grid nodes in D is typically adequate, since the macrostate ρ varies on the large scale. In order to accommodate some extreme cases in which ρ varies more rapidly, more refined grids are used as necessary in the implemented computations discussed below. Even with this finer resolution, the method produces a solution in the order of 100 seconds on a SPARC2 workstation.

5. Computed results.

5.1. Branches of equilibria parametrized by energy. The energy of a given vorticity distribution is the principal factor that determines how it organizes into a coherent structure [22]. Indeed, the coalescence of vorticity into a coherent vortex, which is a negative temperature macrostate, is expected to occur when the energy is sufficiently large [21]. Moreover, the spatial structure of such a vortex and the variation of the mean vorticity within its core depend upon the value of the energy. Consequently, we devote our first set of results to the computation of maximum entropy solutions parametrized by the value of the energy constraint.

We solve the statistical equilibrium problem (2.18) in the shear layer configuration on the fundamental domain $D = \{0 < x_1 < 1, -0.5 < x_2 < 0.5\}$, discretized with 61 grid nodes in both the x_1 and x_2 directions. We normalize the circulation to $C_0 = 1$ and we set $\lambda = 10$ with the consequence that $\rho_{\text{hom}} = 0.1$. We restrict our attention to solutions that are symmetric with respect to x_2 in the sense that $\rho(x_1, x_2) = \rho(x_1, -x_2)$. This restriction allows us to ignore the linear impulse constraint M and its multiplier γ since both are necessarily zero. We enforce this symmetry in our computations by choosing a symmetric initialization for the iterative algorithm, which preserves the symmetry at each iteration.

In Figure 1, the branch of macrostates with E_0 greater than E_{hom} is displayed. This figure shows the level curves of the local probability ρ at increments of 0.1; the figures that follow use the same convention. The first member of the branch, which has an energy value of $E_0 = 0.1350$, is computed by initializing the algorithm with a centered, circular patch whose radius is chosen to satisfy the circulation constraint. Since its energy is very close to $E_{\max}$, it corresponds to a large negative value of $\beta = -15.2$ and exhibits a steep transition between the vortex core ($\rho \approx 1$) and the surrounding irrotational region ($\rho \approx 0$). Larger feasible values of E_0 give equilibria that are very close to steady vortex patches, the deterministic limiting solutions. The subsequent members of the branch displayed in Figure 1 have smaller values of E_0, with correspondingly smaller values of $|\beta|$. These solutions are computed by using a continuation

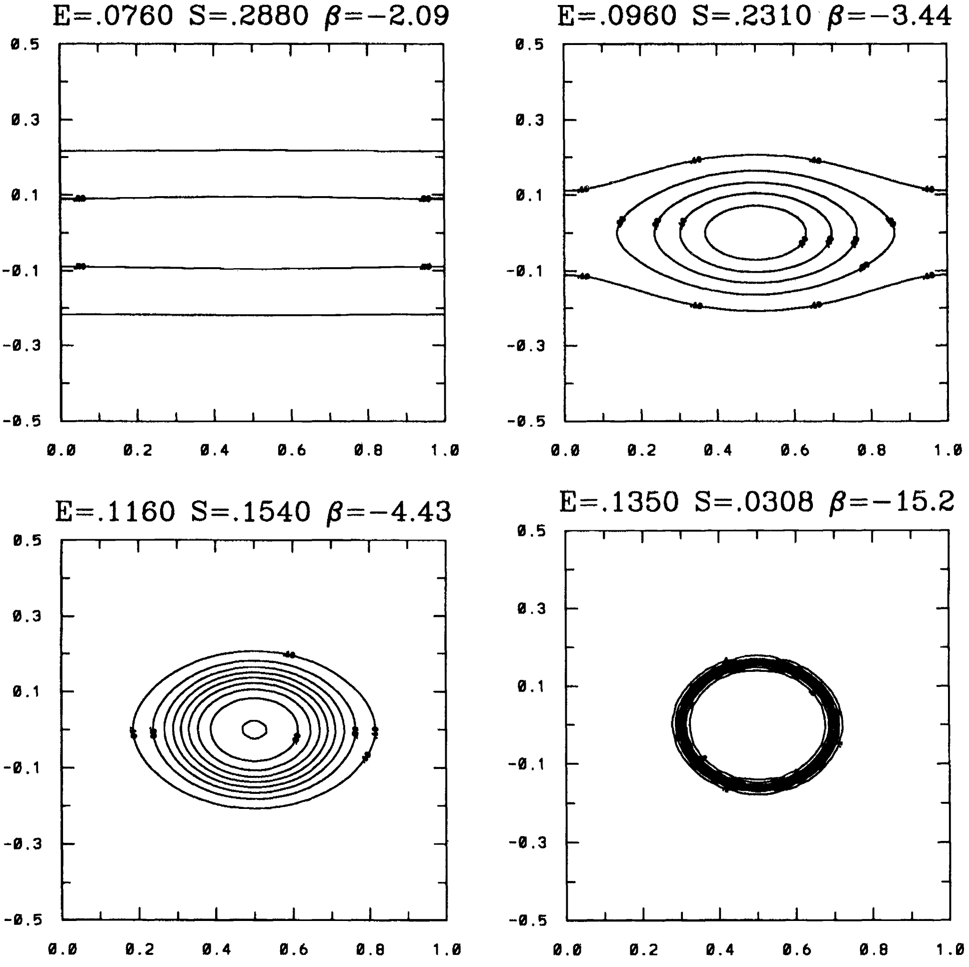
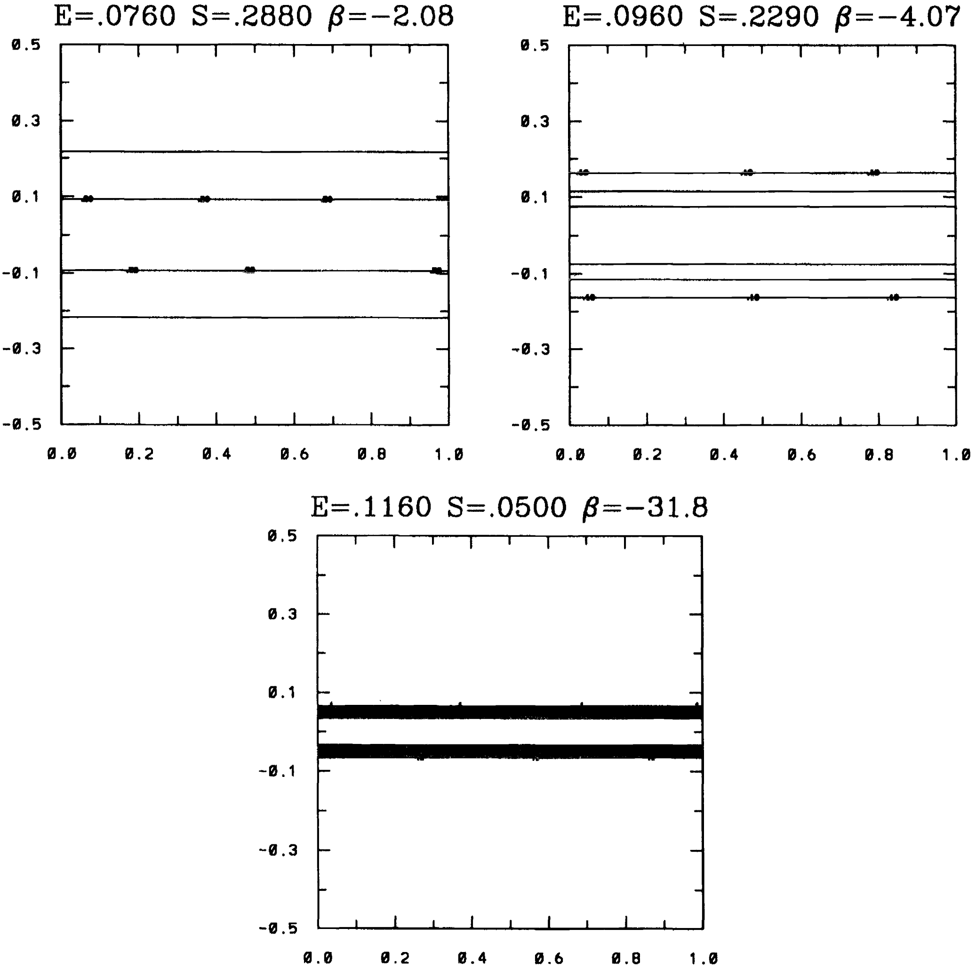


FIG. 1. *Mean streamlines for a branch of maximum entropy solutions with circulation* $C = 1$, *vorticity strength* $\lambda = 10$, *and displayed energy values in the range* $E_{\text{hom}} < E_0 < E_{\max}$ *and computed values of entropy* S *and inverse temperature* β.

strategy in which the algorithm is initialized by a previous solution along the branch having greater energy. As E_0 decreases, the maximum of ρ decreases, indicating greater mixing of entrained irrotational fluid into the vortex core. At the nearly critical value of $E_0 = 0.0760$ the mean vorticity is almost x_1-independent, which evidences almost complete homogeneity in the x_1 direction. Below a critical value of E_0 slightly less than 0.0760, the branch becomes exactly x_1-independent. With decreasing E_0, the x_2 dependence of these solutions decays until the completely homogenous (x-independent) macrostate is reached at E_{hom}.

If the x_1-independence of solutions is imposed, then the above branch of equilibrium shear layers can be continued to a maximum energy limit. This is easily accomplished by initializing the algorithm with an x_1-independent ρ^0 since the x_1 symmetry is preserved at each iteration. Such a branch is displayed in Figure 2. The x_2-dependence of these solutions grows until the parallel vortex patch is reached at a maximum energy slightly greater than $E_0 = 0.1160$. It is interesting to compare the entropies of the solutions in Figure 1 with those in Figure 2 having the same values of the constraints E_0 and C_0. For instance, for $E_0 = 0.0960$ the coherent vortex in Figure 1 has the entropy value $S = 0.2310$ while the parallel shear layer in Figure 2 has the smaller value $S = 0.2290$. This may be interpreted as indicating that the vortex is the most probable state into which the shear layer will evolve if its x_1-translational symmetry


E=.0760 S=.2880 β=−2.08

E=.0960 S=.2290 β=−4.07

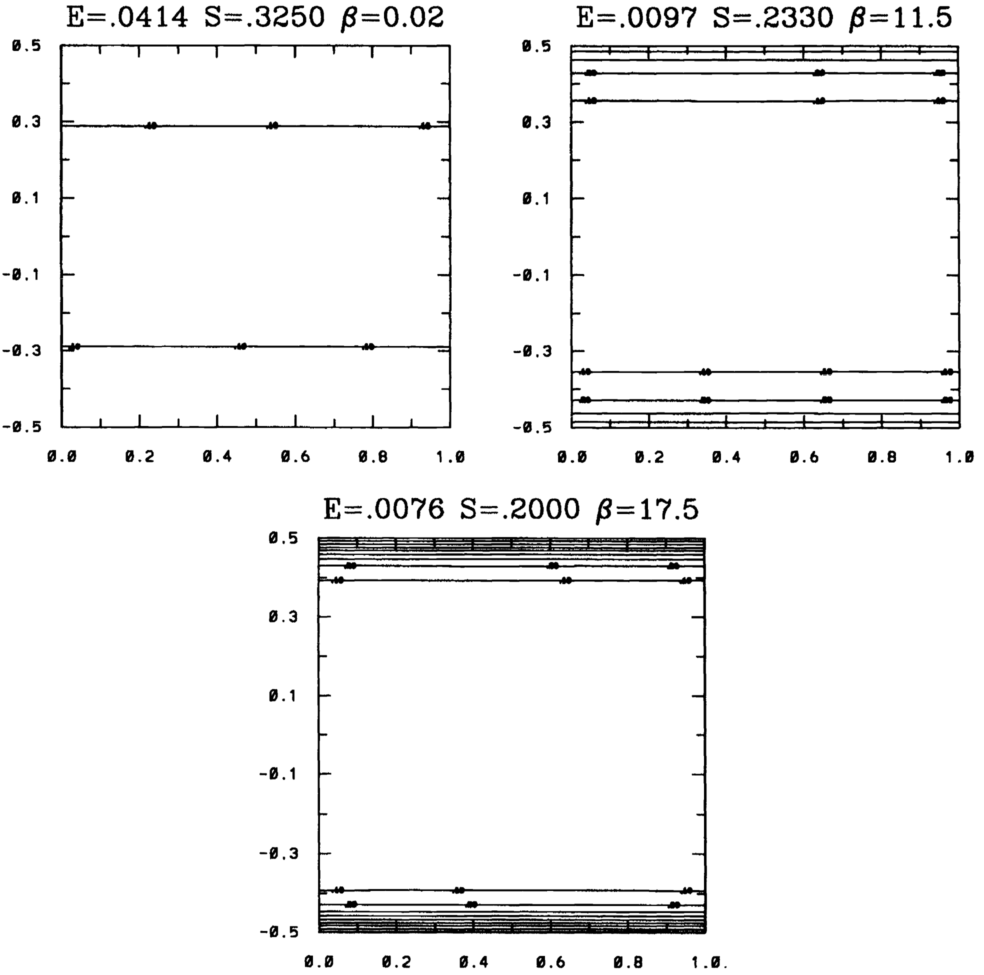
E=.1160 S=.0500 β=−31.8

FIG. 2. *Mean streamlines for a branch of x_1-independent solutions with circulation $C = 1$, vorticity strength $\lambda = 10$, and displayed energy values in the range $E_{\text{hom}} < E_0 < E_{\max}$ and computed values of entropy S and inverse temperature β.*

is broken. These computations therefore provide a confirmation of the rollup phenomenon exhibited in dynamical evolution of unstable shear layers.

The continuation of these coincident branches to energy values less than E_{hom} is given in Figure 3. For these values of E_0, β is positive and the solutions are x_1-independent. As E_0 decreases along this branch of solutions, β increases and the vorticity accumulates toward the walls. This behavior is anticipated on heuristic grounds. Rather curiously, our algorithm converges to solutions on this branch even though our convergence proofs depend on the strict negativity of β.

In Figure 4, we plot the dependence of the equilibrium value of S upon the given value of E_0 for the full feasible range of the energy and for both the x_1-dependent and x_1-independent branches. This graph demonstrates how maximum entropy coherent vortex structures bifurcate from parallel shear layers as their energy is increased. Since the computation of these branches is accomplished by our iterative algorithm which is entropy increasing, we find only those branches whose members are entropy maximizing, possibly subject to a symmetry restriction. We believe that the x_1-dependent branch displayed here is the absolute maximum entropy branch.


E=.0414 S=.3250 β=0.02

E=.0097 S=.2330 β=11.5

E=.0076 S=.2000 β=17.5

FIG. 3. *Branch of solutions with circulation $C = 1$, vorticity strength $\lambda = 10$, and displayed energy values in the range $E_{\min} < E_0 < E_{\text{hom}}$ and computed values of entropy S and inverse temperature β.*

5.2. Rollup of shear layers. In the context of x_1-periodic, x_2-bounded flows, the most basic prediction of the statistical equilibrium theory resides in the correspondence between an initial perturbed shear layer and a final array of coherent vortices. In particular, the computational solution of the maximum entropy principle (2.18) establishes a definite, quantitative relationship between the thickness of the given shear layer and the form of the coherent structure into which it evolves. We examine this relationship in our second set of computations.

We now take a domain $D = \{0 < x_1 < 1\,, -1 < x_2 < 1\}$, which we discretize with 81 grid nodes in each direction. With $C = 1$ as before, we determine the vorticity strength λ and the energy E_0 from a shear layer that is in the form of a parallel vortex patch: $\omega_0 = \lambda$ in $\{|x_2| < \delta/2\}$, $\omega_0 = 0$ elsewhere. The normalization of circulation obviously implies that $\lambda = \delta^{-1}$. Consequently, the thickness δ of the shear layer completely determines the constraints in (2.18). The energy value E_0 is a decreasing function of δ and takes its largest value in the limit as $\delta \to 0$, which corresponds to a (unit strength) vortex sheet. For $\delta \ll 1$, this configuration is a good approximation to a free shear layer with a linear velocity profile, the bounding walls at $x_2 = \pm 1$ being sufficiently far from the layer centered at $x_2 = 0$ to have only a small influence.

We compute a branch of equilibria parametrized by δ, using an initialization that breaks the x_1-independence of the defining shear layer. The iterative algorithm then mimics the

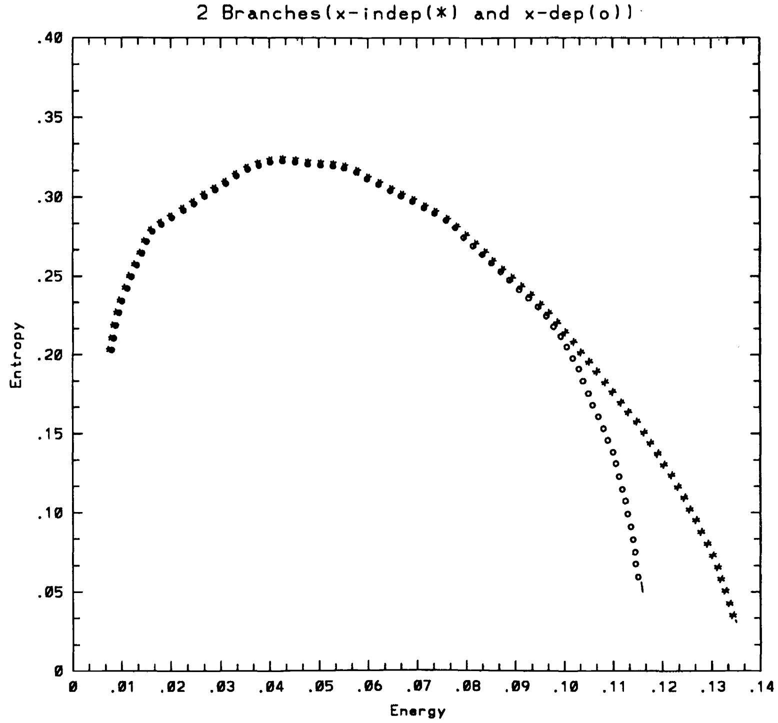


FIG. 4. *The dependence of the equilibrium value of S on the constraint value E_0 over the full feasible range. The branch of x_1-dependent solutions (coherent vortices) bifurcates from the branch of x_1-independent solutions (shear layers).*

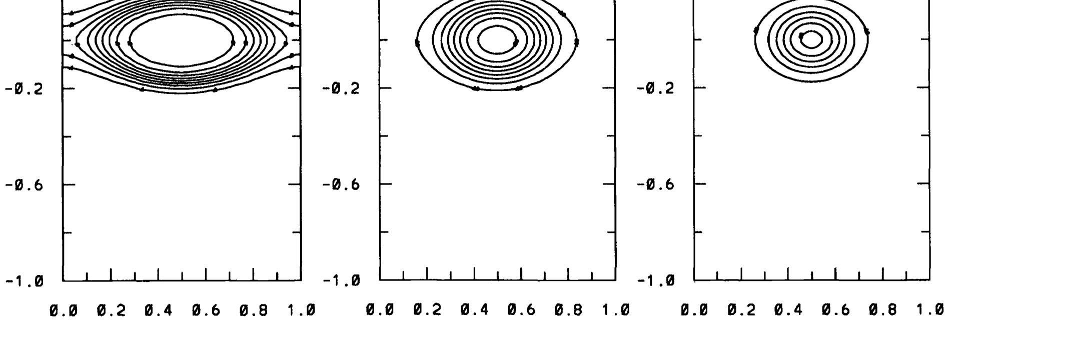


FIG. 5. *Rollup of shear layers with thicknesses δ into coherent vortices.*

temporal evolution of the perturbed shear layer as it rolls up into a coherent vortex, which the statistical equilibrium model takes to be the maximum entropy solution. In Figure 5, we show these solutions for layer thicknesses of $\delta = 0.2, 0.1, 0.05$. As δ is decreased, the vortex structure develops a more pronounced core and the mean vorticity becomes more concentrated

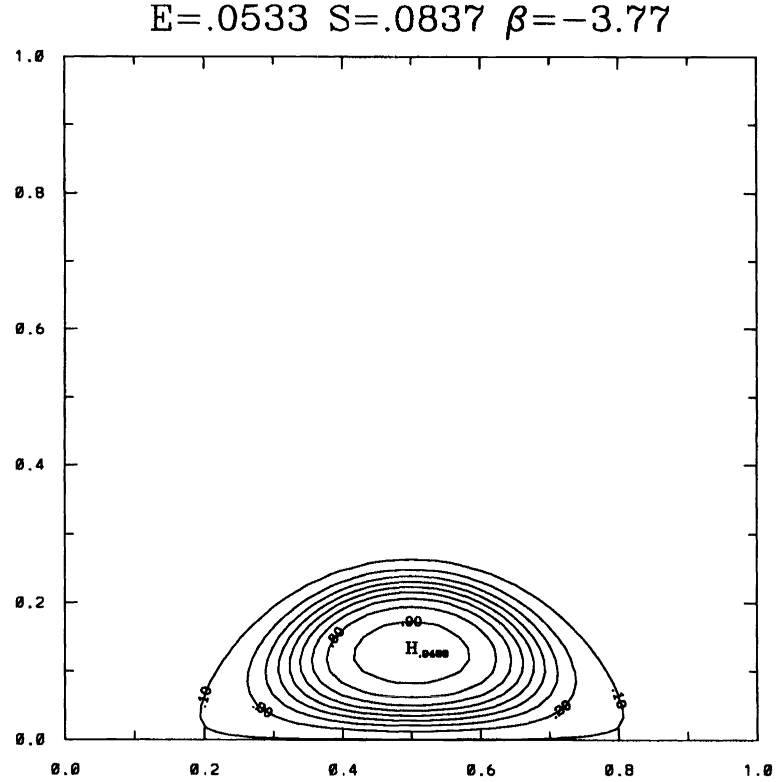


FIG. 6. *Coalescence of two circular vortex patches into a coherent vortex.*

in that core. Nevertheless, the maximum of the volume fraction ρ (which occurs at the center of the vortex) decreases along with δ, indicating that the thinner shear layers result in coherent vortices with greater mixing in their cores. This behavior is in good qualitative agreement with the results of direct simulation of rollup in thin shear layers [10] and (desingularized) vortex sheets [2, 15]. A further quantitative comparison with such results would certainly be very interesting but will not be included here.

5.3. Coalescence of vortices. A common feature of shear layer turbulence is the pairing or merging of like-signed regions of vorticity. As a simple example of this general phenomenon, we consider the leap-frogging behavior of two identical vortex patches, a combination of translational and corotational motion. The statistical equilibrium theory predicts that the longtime average of this evolution leads to a steadily translating, coalesced vortex structure. We take this example as the third application of our numerical method. In contrast to the two preceding cases in which the multiplier γ (translational speed) can be ignored, the impulse constraint is active in this case.

We are interested in the behavior of two vortex patches of unit circulation in the fundamental domain $D = \{0 < x_1 < 1,\ 0 < x_2 < 1\}$. Initially the patches are circular with centers at (0.35, 0.125) and (0.65, 0.125), and both radii are equal to 0.1. This distribution ω_0 determines the constraint values C_0, E_0, M_0 in the maximum entropy principle. Since the vortices are close to the wall at $x_2 = 0$ and are closer to each other in x_1 separation than to their periodic images, they are expected to translate in the positive x_1 direction and to corotate mutually. The distortion caused by the boundary wall and the mutual straining then promotes the coalescence of the vortex patches. The resulting coherent structure is modeled by the maximum entropy solution displayed in Figure 6. It has a translational speed of $-\gamma/\beta$, with computed values of $\beta = -3.77$ and $\gamma = 1.92$. In addition to the form of this macroscopic vortex structure, the statistical equilibrium theory thus predicts its speed of translation, using the conservation of linear impulse associated with the underlying dynamics.

Acknowledgment. The authors express their sincere thanks to Richard Jordan for his many valuable comments and illuminating conversations.

REFERENCES

[1] G. K. Batchelor, *An Introduction to Fluid Dynamics*, Cambridge University Press, Cambridge, 1967.

[2] G. R. Baker and M. J. Shelley, *On the connection between thin vortex layers and vortex sheets*, J. Fluid Mech., 215 (1990), p. 161.

[3] M. E. Brachet, M. Meneguzzi, H. Politano, and P. L. Sulem, *The dynamics of freely decaying two-dimensional turbulence*, J. Fluid Mech., 194 (1988), p. 333.

[4] E. Cagloti, P. L. Lions, C. Marchioro, and M. Pulvirenti, *A special class of stationary flows for two-dimensional Euler equations: A statistical mechanics description*, Comm. Math. Phys., 143 (1992), p. 501.

[5] A. Chorin, *Statistical mechanics and vortex motion*, in Lectures in Appl. Math., 28, American Mathematical Society, Providence, RI, 1991, p. 85.

[6] D. Dritschel, *Contour surgery: A topological reconnection scheme for extended integrations using contour dynamics*, J. Comput. Phys., 77 (1988), p. 240.

[7] ———, *The repeated filamentation of two-dimensional vorticity interfaces,* J. Fluid Mech., 194 (1988), p. 511.

[8] L. C. Evans, *Weak Convergence Methods for Nonlinear Partial Differential Equations*, CBMS Regional Conference Series in Mathematics, 74, American Mathematical Society, Providence, RI, 1990.

[9] A. Eydeland and B. Turkington, *A computational method of solving free-boundary problems in vortex dynamics*, J. Comput. Phys., 78 (1988), p. 194.

[10] A. F. Ghoniem, G. Heidarinejad, and A. Krishnan, *Numerical simulation of a thermally stratified shear layer using the vortex element method*, J. Comput. Phys., 79 (1988), p. 135.

[11] A. D. Ioffe and V. M. Tihomirov, *Theory of Extremal Problems*, Elsevier–North Holland, New York, 1979.

[12] T. Kato, *On the classical solutions of the two-dimensional non stationary Euler equations*, Arch. Rational Mech. Anal., 25 (1967), p. 302.

[13] M. Kiessling, *Statistical mechanics of classical particles with logarithmic interactions*, Comm. Pure Appl. Math., 46 (1993), p. 27.

[14] R. Kraichnan and D. Montgomery, *Two-dimensional turbulence*, Rep. Progr. Phys., 43 (1980), p. 547.

[15] R. Krasny, *Desingularization of periodic vortex sheet roll-up*, J. Comput. Phys., 65 (1986), p. 292.

[16] T. S. Lundgren and Y. B. Pointin, *Statistical mechanics of two-dimensional vortices*, J. Statist. Phys., 17 (1977), p. 323.

[17] J. C. McWilliams, *The emergence of isolated coherent vortices in turbulent flow*, J. Fluid Mech., 146 (1984), p. 21.

[18] J. Miller, *Statistical mechanics of Euler equations in two dimensions*, Phys. Rev. Lett., 65 (1990), p. 2137.

[19] J. Miller, P. Weichman, and M. C. Cross, *Statistical mechanics, Euler's equations, and Jupiter's red spot*, Phys. Rev. A, 45 (1992), p. 2328.

[20] D. Montgomery and G. Joyce, *Statistical mechanics of "negative temperature" states*, Phys. Fluids, 17 (1974), p. 1139.

[21] L. Onsager, *Statistical hydrodynamics*, Suppl. Nuovo Cim., 6 (1949), p. 279.

[22] R. T. Pierrehumbert and S. E. Widnall, *The structure of organized vortices in a free shear layer*, J. Fluid Mech., 102 (1981), p. 301.

[23] R. Robert, *A maximum-entropy principle for two-dimensional perfect fluid dynamics*, J. Statist. Phys., 65 (1991), p. 531.

[24] R. Robert and J. Sommeria, *Statistical equilibrium states for two-dimensional flows*, J. Fluid Mech., 229 (1991), p. 291.

[25] ———, *Relaxation towards a statistical equilibrium state in two-dimensional perfect fluid dynamics*, Phys. Rev. Lett., 69 (1992), p. 2776.

[26] R. T. Rockafellar, *Convex Analysis*, Princeton University Press, Princeton, NJ, 1970.

[27] J. Sommeria, C. Staquet, and R. Robert, *Final equilibrium state of a two-dimensional shear layer*, J. Fluid Mech., 233 (1991), p. 661.

[28] B. Turkington, *On steady vortex flow in two dimensions.* I, Comm. Partial Differential Equations, 8 (1983), p. 999.

[29] V. I. Youdovitch, *Non stationary flow of an ideal incompressible liquid*, Zh. Vychisl. Mati. Mat. Fiz., 3 (1963), p. 1032.

[30] N. J. Zabusky, M. H. Hughes, and K. V. Roberts, *Contour dynamics for the Euler equations in two dimensions,* J. Comput. Phys., 30 (1979), p. 96.

SIAM J. SCI. COMPUT.
Vol. 17, No. 6, pp. 1434–1448, November 1996

SPACE–TIME FINITE ELEMENT METHODS FOR SURFACE DIFFUSION WITH APPLICATIONS TO THE THEORY OF THE STABILITY OF CYLINDERS*

BERNARD D. COLEMAN†, RICHARD S. FALK†, AND MAHER MOAKHER†

Abstract. A family of space–time finite element approximation schemes is presented for the nonlinear partial differential equations governing diffusion in the surface of a body of revolution. The schemes share with the partial differential equations properties of conservation of volume and decrease of area. Numerical experiments are described showing that the result of the linear theory of small amplitude longitudinal perturbations of a cylinder to the effect that a long cylinder is stable against all perturbations with spatial Fourier spectra containing only wavelengths less than the circumference of the cylinder does not hold in the full nonlinear theory. Examples are given of cases in which longitudinal perturbations with high wave-number spectra grow in amplitude, after an initial rapid decay followed by a long "incubation period," and result in break-up of the body into a necklace of beads. The results of finite element calculations are compared with the predictions of a perturbation analysis.

Key words. axially symmetric motion by Laplacian of mean curvature, stability against surface diffusion

AMS subject classifications. 65M60, 73V05, 73T05

1. Introduction.

We are concerned here with the numerical computation of morphological changes induced in an isotropic and homogeneous solid body by mass diffusion within the body's bounding surface $\mathcal{S}$. We employ a constitutive equation, due to Herring [6], expressing the mass flux $\boldsymbol{q}$ in $\mathcal{S}$ as a linear function of the gradient in $\mathcal{S}$ of the sum H of the principal curvatures:

$$\boldsymbol{q} = -K\nabla_s H. \tag{1.1}$$

Here $K > 0$ is a material constant proportional to the surface self-diffusion coefficient of the isotropic material of which the body is composed, and we are using a sign convention for curvature such that H is positive for a sphere. As was observed and exploited by Mullins [9], when the only motion is the flux $\boldsymbol{q}$ in $\mathcal{S}$, the mass balance yields the following relation between the rate v of advance of $\mathcal{S}$ along its exterior normal and the surface divergence of $\boldsymbol{q}$:

$$\rho v + \mathrm{div}_s\boldsymbol{q} = 0; \tag{1.2}$$

here ρ is the mass density, per unit volume, of the material in the body.

For a given characteristic length L, the theory of (1.1), (1.2) is rendered dimensionless by replacing quantities x, r, etc., with dimension of length by xL, rL, etc., the time t by $\rho L^4 t/K$, and hence $\boldsymbol{q}$ by $K\boldsymbol{q}/L^2$ and H by H/L. When this is done, (1.1), (1.2) yield

$$v = \Delta_s H, \tag{1.3}$$

and thus are said to govern the theory of *motion by Laplacian of mean curvature*. That theory has a less-developed literature than the theory of *motion by mean curvature*, which is based on the equation

$$v = -H. \tag{1.4}$$

*Received by the editors September 21, 1994; accepted for publication (in revised form) June 16, 1995. This research was supported by National Science Foundation grants DMS-94-04580 and DMS-94-03552 and by the Donors of the Petroleum Research Fund, administered by the American Chemical Society.

†Department of Mechanics and Materials Science, and Department of Mathematics, Rutgers University, Piscataway, NJ 08854 (bcoleman@duhem.rutgers.edu, falk@math.rutgers.edu, moakher@caip.rutgers.edu).

In a recent survey [1], Cahn and Taylor discuss the difficulties encountered when one attempts to extend techniques employed to develop the theory of motion by mean curvature to the theory of motion by Laplacian of mean curvature. Whereas H in (1.4) is given by second-order spatial derivatives of surface coordinates, $\Delta_s H$ in (1.3) depends on fourth-order derivatives of the coordinates, and this elementary fact has the important consequence that a maximum principle employed in the theory of (1.4) does not hold for (1.3).

Among the papers presenting computational methods for motion by mean curvature are that of Dziuk [3], giving a semidiscretization scheme based on tangential gradients, and that of Sethian [11], based on a level set formulation of equation (1.4).

Among recent analytical developments in the analysis of the motion of surfaces by mean curvature are Huisken's short-time existence and regularity results [8], Soner and Songanidis's analysis of the nature of singularities in axially symmetric surfaces [12], and Evans and Spruck's theory of viscosity solutions [5] for a level set formulation of (1.4).

Interesting and very recent results in the theory of motion by Laplacian of mean curvature in a plane are proofs by Elliott and Garcke [4] of the asymptotic stability of circles and of global existence of planar solutions with initial data in a neighborhood of a circle.

The numerical methods and results presented here are for initial-value problems arising in the theory of motion by Laplacian of mean curvature for axially symmetric surfaces subject to periodic boundary conditions. In §2 we give various forms taken by the system (1.1), (1.2) for such surfaces and discuss relevant laws of conservation of volume and decrease of area [2]. A variational formulation presented in §3 is discretized in §4 with space–time finite element schemes that yield analogues of the laws of conservation of volume and decrease of area that hold for (1.1), (1.2). Results of numerical experiments are presented in §5.

For an axially symmetric surface, in a natural cylindrical coordinate system with radial coordinate r and axial coordinate x, the equation (1.3) of motion by Laplacian of mean curvature becomes

$$r_t = \frac{1}{r}\left[\frac{r}{(1+r_x^2)^{1/2}}\left(\frac{1}{r(1+r_x^2)^{1/2}} - \frac{r_{xx}}{(1+r_x^2)^{3/2}}\right)_x\right]_x . \tag{1.5}$$

The emphasis here is on cases in which this equation is subject to initial data of the form

$$r(x,0) = r_0(x) = 1 + \epsilon u(x) \tag{1.6}$$

with $\epsilon > 0$ and u an almost periodic function. For simplicity we often take u to have a finite Fourier spectrum, i.e., to be of the form

$$u(x) = \sum_{i=1}^{M} c_i \sin(k_i x + \varphi_i) \tag{1.7}$$

with $c_i,\ k_i > 0$. (In the numerical calculations of this paper u is assumed periodic and hence the wave numbers k_i in (1.7) are commensurate.) For longitudinal perturbations of a cylinder of radius a, (1.5) is obtained from (1.1), (1.2) by putting $L = a$ and making the change of units discussed above. For a given u, the perturbation is small if ϵ is small compared with 1, and, in (1.7), $k_i < 1$, $= 1$, or > 1, in accord with whether the corresponding period $P_i = 2\pi/k_i$ exceeds, equals, or is less than the circumference of the unperturbed cylinder.

Since the work of Nichols and Mullins [10], in the study of small amplitude longitudinal perturbations of a cylinder it has been customary to restrict attention to the linearization of (1.5) about $r = 1$, i.e., to the equation

$$r_t + r_{xxxx} + r_{xx} = 0, \tag{1.8}$$

whose solution with the initial condition (1.6), (1.7) is

$$r(x,t) = 1 + \epsilon \sum_{i=1}^{M} c_i e^{\alpha(k_i)t} \sin(k_i x + \varphi_i) \tag{1.9}$$

where α obeys the dispersion relation

$$\alpha(k) = k^2(1-k^2). \tag{1.10}$$

The maximum value of $\alpha(k)$ is $\frac{1}{4}$ and occurs at $k = 1/\sqrt{2}$. As the right side of (1.10) is negative for $k > 1$, the linear equation (1.8) implies that whenever, in (1.7), $k_i > 1$ for all i, the perturbation decays to zero as $t \to \infty$. We recently presented arguments [2] to the effect that such is not the case in the theory of the nonlinear equation (1.5).

The arguments given in [2] are based on formal perturbation analyses employing an expansion of the solution of (1.5), (1.6) in ϵ. A perturbation argument taking into account terms $O(\epsilon^2)$ yields the conclusion that if

$$k_i > 1 \qquad \text{for} \qquad i = 1, \dots, M \tag{1.11a}$$

and, in addition, two distinct wave numbers, k_i, k_j, have $|k_i - k_j| < 1$, then, although the solution of (1.5) will exhibit an initial decay of the perturbation, after a time whose duration can be estimated, a new sinusoidal term with wave number equal to $|k_i - k_j|$, i.e., to that of the envelope of the ith and jth terms in (1.7), will appear in the solution and grow in amplitude, taking the surface far from cylindrical shape. When terms $O(\epsilon^n)$ are taken into account the following generalization of this conclusion is obtained: if (1.11a) holds, and, in addition, there is an integer $n \geq 2$ and there are M integers m_i (positive, negative, or zero) that obey the relations

$$\sum_{i=1}^{M} |m_i| = n \qquad \text{and} \qquad 0 < \left| \sum_{i=1}^{M} m_i k_i \right| < 1, \tag{1.11b}$$

then, again after an initial decay, the perturbation will grow.

The perturbation analysis given in [2] and discussed here in §5 leads one to expect, but cannot be employed to prove, that in the cases in which (1.11a) and (1.11b) hold the growth in perturbations proceeds until there is a time t^* and values x^* of x for which $\lim_{t \to t^*} r(x^*, t) = 0$. The numerical methods we present here were developed to see if such is the case and, if so, to permit precise calculation of the break-up time t^* as well as the "incubation time" required for occurrence of appreciable growth of a perturbation after its initial (and generally rapid) decay. Numerical experiments confirming expectations based on the perturbation analysis are described in §5. We also give there an example of a case in which our numerical methods can be applied to study the evolution of the body for $t > t^*$. In general, a topological change occurs at $t = t^*$, resulting in the break-up of the original body into separated subbodies. Thus we refer to t^* as the "time of break-up." (In the theory of motion by mean curvature, in which the volume is not conserved but decreases in time, similar phenomena can occur, and their times of occurrence are often referred to as times of "pinch-off.")

2. Basic equations.

To describe the evolution of the surface of an infinite body of revolution, we continue to use the cylindrical coordinates x and r of equation (1.5). We write $\boldsymbol{t}$ for the unit tangent to the time-dependent curve $r = r(x,t)$ in the (x,r)-plane, and we define the signed magnitude q of $\boldsymbol{q}$ so that $\boldsymbol{q} = q\boldsymbol{t}$. For axially symmetric surfaces, when the dimensionless units are used, (1.2) and (1.1) become, respectively,

$$rr_t = -(rq)_x, \tag{2.1}$$

$$q = -(\nabla_s H) \cdot \boldsymbol{t} = -\frac{H_x}{\sqrt{1+r_x^2}} \tag{2.2}$$

where

$$H = \frac{1}{r}\left[\sqrt{1+r_x^2} - \left(\frac{rr_x}{\sqrt{1+r_x^2}}\right)_x\right]. \tag{2.3}$$

For numerical calculations we treat the body as one with finite length subject to space-periodic boundary conditions with, say, period P. The resulting problems have properties of volume conservation and area decay which we now discuss.

It follows from (2.1) that the volume of material lying between cross-sectional planes $x = a$ and $x = b$ with $b - a = P$, i.e.,

$$V(t) = \pi \int_a^b r^2\, dx, \tag{2.4}$$

is constant in time if r and $q = -H_x/\sqrt{1+r_x^2}$ are P-periodic. Equations (2.1) and (2.2) together imply that the surface area (or the free energy) of the portion of the body lying between these two planes, i.e.,

$$\Psi(t) = 2\pi \int_a^b r\sqrt{1+r_x^2}\, dx, \tag{2.5}$$

is a monotone-decreasing function of time, provided that not only r and $H_x/\sqrt{1+r_x^2}$ but also H and $r_x/\sqrt{1+r_x^2}$ are P-periodic. The periodicity of r, H, $r_x/\sqrt{1+r_x^2}$, and $H_x/\sqrt{1+r_x^2}$ are equivalent to the periodicity of r, H, r_x, and H_x. Thus, under boundary conditions implying such periodicity, V is preserved and Ψ represents a Lyapunov function for the evolution of r.

We put $\Omega = (a, b)$ and write $\mathcal{I} = (0, T)$ for an interval with $T > 0$ on which solutions of (2.1)–(2.3) with $r > 0$ are defined. Under the conditions of periodicity just described

$$\dot{\Psi} = -2\pi \int_\Omega \frac{rH_x^2}{\sqrt{1+r_x^2}} dx = -2\pi \int_\Omega q^2 r\sqrt{1+r_x^2} dx \leq 0, \tag{2.6}$$

and hence under the same conditions

$$\Phi(t) = \Psi(t) + 2\pi \int_0^t \int_\Omega q^2(x,\tau) r(x,\tau)\sqrt{1+r_x^2(x,\tau)}\, dx\, d\tau \tag{2.7}$$

is constant:

$$\Phi(t) = \Phi(0) = \Psi(0). \tag{2.8}$$

For each initial configuration r_0 we seek $r(\cdot, t)$ satisfying (2.1), (2.2), and the periodicity conditions. The equation of evolution for r, (1.5), which is fourth order in the space variable x, can be formulated in terms of r and H to yield a coupled system of second order in x. The resulting fully nonlinear initial-value problem with periodic boundary conditions has a variational formulation that can be slightly simplified by introducing a function R defined by

$$R(x,t) = \frac{1}{2} r^2(x,t). \tag{2.9}$$

In this way we are led to the following problem.

Problem RH. Find (R, H) with $R > 0$ satisfying

$$R_t = \left(\frac{2RH_x}{\sqrt{2R + R_x^2}}\right)_x \qquad \forall (x, t) \in \Omega \times \mathcal{I}, \tag{2.10}$$

$$H = \frac{1}{\sqrt{2R + R_x^2}} - \left(\frac{R_x}{\sqrt{2R + R_x^2}}\right)_x \qquad \forall (x, t) \in \Omega \times \mathcal{I}, \tag{2.11}$$

with the periodic boundary conditions

$$\begin{aligned} R(a, t) &= R(b, t), \qquad R_x(a, t) = R_x(b, t) \qquad \forall t \in \mathcal{I}, \\ H(a, t) &= H(b, t), \qquad H_x(a, t) = H_x(b, t) \qquad \forall t \in \mathcal{I}, \end{aligned} \tag{2.12}$$

and the initial condition

$$R(x, 0) = R_0(x) = \frac{1}{2} r_0^2(x) \qquad \forall x \in \Omega. \tag{2.13}$$

3. Variational formulation. We use standard notation: $L^2(\Omega)$ is the space of square integrable functions on Ω, $L^\infty(\Omega)$ is the space of essentially bounded functions on Ω, $H^1(\Omega)$ is the Sobolev space of functions in $L^2(\Omega)$ with distributional derivative in $L^2(\Omega)$, and $W^{1,\infty}(\Omega)$ is the Sobolev space of functions in $L^\infty(\Omega)$ with distributional derivative in $L^\infty(\Omega)$. Of importance here are subspaces H_{P}^1 and $W_{\mathrm{P}}^{1,\infty}$ of $H^1(\Omega)$ and $W^{1,\infty}(\Omega)$, with $\Omega = (a, b)$, that arise from our concern with periodic boundary conditions

$$\begin{aligned} H_{\mathrm{P}}^1 &= \left\{ v \in H^1(\Omega) : v(a) = v(b) \right\}, \\ W_{\mathrm{P}}^{1,\infty} &= \left\{ v \in W^{1,\infty}(\Omega) : v(a) = v(b) \right\}. \end{aligned}$$

We also define H_{P}^{-1} as the dual space of H_{P}^1. Let X be a Banach space on Ω with norm $|| \cdot ||_X$. We denote by $L^2(0, T; X)$ the space of functions f from $(0, T)$ into X such that $(\int_0^T ||f||_X^2 dt)^{1/2} < \infty$ and by $L^\infty(0, T; X)$ the space of functions f from $(0, T)$ into X such that $||f||_X$ is essentially bounded on $(0, T)$.

For brevity we write

$$(f, g) = \int_\Omega f g \, dx.$$

Variational form of Problem RH. Find $R \in L^\infty(0, T; W_{\mathrm{P}}^{1,\infty})$ with $R > 0$ and $R_t \in L^2(0, T; H_{\mathrm{P}}^{-1})$ and $H \in L^2(0, T; H_{\mathrm{P}}^1)$ satisfying

$$\int_0^T (R_t, u) \, dt + \int_0^T \left(\frac{2RH_x}{\sqrt{2R + R_x^2}}, u_x \right) dt = 0 \quad \forall u \in L^2(0, T; H_{\mathrm{P}}^1), \tag{3.1}$$

$$\int_0^T (H, v) \, dt = \int_0^T \left(\frac{1}{\sqrt{2R + R_x^2}}, v \right) dt + \int_0^T \left(\frac{R_x}{\sqrt{2R + R_x^2}}, v_x \right) dt \quad \forall v \in L^2(0, T; H_{\mathrm{P}}^1), \tag{3.2}$$

$$(R(\cdot, 0), w) = (R_0, w) \quad \forall w \in L^2(\Omega). \tag{3.3}$$

We note that there are not yet available proofs of existence and uniqueness for solutions for either Problem RH or (3.1)–(3.3). However, if we are granted existence and uniqueness, the stated variational form of Problem RH is a useful formulation for the construction of the conforming finite element approximation schemes presented in §4.

Straightforward arguments show that solutions of (2.10)–(2.13) obey (3.1)–(3.3) and that solutions of the variational equations (3.1)–(3.3) with sufficient regularity obey (2.10)–(2.13).

Constancy of volume and monotone decrease of surface area hold for each solution (R, H) of (3.1)–(3.3). To show this, we observe that in our present notation

$$\dot{V} = 2\pi(R_t, 1), \tag{3.4}$$

and hence by putting $u \equiv 1$ in the time interval $(0, t)$ and $u \equiv 0$ in the time interval (t, T) in equation (3.1), we obtain

$$V(t) = V(0). \tag{3.5}$$

In (3.1) and (3.2) by putting $u = H$ and $v = R_t$ in the time interval $(0, t)$ and $u \equiv 0$, $v \equiv 0$ in the time interval (t, T), both of which are admissible test functions, we obtain

$$\int_0^t (R_t, H)\, dt + \int_0^t \left(\frac{2R}{\sqrt{2R + R_x^2}}, H_x^2 \right) dt = 0, \tag{3.6}$$

$$\int_0^t (H, R_t)\, dt = \int_0^t \left(\frac{1}{\sqrt{2R + R_x^2}}, R_t \right) dt + \int_0^t \left(\frac{R_x}{\sqrt{2R + R_x^2}}, R_{tx} \right) dt. \tag{3.7}$$

As $\Psi = 2\pi(\sqrt{2R + R_x^2}, 1)$ and hence

$$\dot{\Psi} = 2\pi \left(\frac{1}{\sqrt{2R + R_x^2}}, R_t \right) + 2\pi \left(\frac{R_x}{\sqrt{2R + R_x^2}}, R_{tx} \right), \tag{3.8}$$

(3.7) yields $\int_0^t \dot{\Psi}\, dt = 2\pi \int_0^t (H, R_t)\, dt$, and, by (3.6),

$$\Psi(t) - \Psi(0) = -2\pi \int_0^t \left(\frac{2R}{\sqrt{2R + R_x^2}}, H_x^2 \right) dt. \tag{3.9}$$

Hence

$$\Psi(t) - \Psi(0) = -2\pi \int_0^t \left(\frac{2R}{\sqrt{2R + R_x^2}}, H_x^2 \right) dt \leq 0.$$

4. Finite element method. To approximate the variational form of Problem RH, we here construct a family of mixed finite element methods for which piecewise polynomial approximations in space and time are used for both R and H. Let $a = x_0 < x_1 < \cdots < x_N = b$ and $0 = t^0 < t^1 < \cdots < t^K = T$ be partitions of Ω and $\mathcal{I}$, and let $h_i = x_{i+1} - x_i$, $i = 0, \ldots, N-1$ and $k^j = t^{j+1} - t^j$, $j = 0, \ldots, K-1$, be mesh spacings and time steps. Let $S_h^p(\Omega)$ be the finite element space of continuous functions Q that are piecewise polynomials of degree $p \geq 1$ on each interval of the partition of Ω and obey the periodicity condition $Q(a) = Q(b)$. We write $S_k^q(\mathcal{I})$ and $\bar{S}_k^q(\mathcal{I})$, respectively, for the finite element spaces of continuous and discontinuous functions that are piecewise polynomials of degree $q \geq 0$ on each interval of the partition of $\mathcal{I}$. The members of the tensor product spaces

$$S_{hk}^{p,q} = S_h^p(\Omega) \otimes S_k^q(\mathcal{I}), \qquad \bar{S}_{hk}^{p,q} = S_h^p(\Omega) \otimes \bar{S}_k^q(\mathcal{I})$$

are functions on $\Omega \times \mathcal{I}$. We consider, for given integers $p \geq 1, q \geq 0$, the following.

Finite element approximation of type (p,q) to Problem RH. Find $R_{hk} \in S_{hk}^{p,q+1}$ and $H_{hk} \in \bar{S}_{hk}^{p,q}$ satisfying

$$\int_0^T ((R_{hk})_t, u_{hk})dt + \int_0^T \left(\frac{2R_{hk}(H_{hk})_x}{\sqrt{2R_{hk} + (R_{hk})_x^2}}, (u_{hk})_x\right) dt = 0 \qquad \forall u_{hk} \in \bar{S}_{hk}^{p,q},$$

$$\int_0^T (H_{hk}, v_{hk})dt = \int_0^T \left(\frac{(R_{hk})_x}{\sqrt{2R_{hk} + (R_{hk})_x^2}}, (v_{hk})_x\right) dt$$

$$+ \int_0^T \left(\frac{1}{\sqrt{2R_{hk} + (R_{hk})_x^2}}, v_{hk}\right) dt \qquad \forall v_{hk} \in \bar{S}_{hk}^{p,q},$$

$$(R_{hk}(\cdot, 0), w_{hk}) = (R_0, w_{hk}) \qquad \forall w_{hk} \in S_h^p(\Omega).$$

We note that R_{hk} is a polynomial of one degree higher in time than H_{hk} and the test functions u_{hk} and v_{hk}. The simplest finite element approximation in this family is that of type (p,q) with $p = 1$ and $q = 0$, i.e., that for which R_{hk} is a continuous piecewise linear polynomial in both space and time and H_{hk} and the test functions u_{hk} and v_{hk} are continuous piecewise linear polynomials in space and piecewise constant in time.

Because the test functions u_{hk} and v_{hk} are discontinuous in time, finite element solutions (R_{hk}, H_{hk}) can be computed by marching through successive time intervals. Let $P^q(\mathcal{I}^n)$ be the set of polynomials on $\mathcal{I}^n = [t^n, t^{n+1}]$ of degree q. On the time-strip $\mathcal{I}^n$ the appropriate restrictions of R_{hk} and H_{hk} belong to $S_h^p(\Omega) \otimes P^{q+1}(\mathcal{I}^n)$ and $S_h^p(\Omega) \otimes P^q(\mathcal{I}^n)$, respectively, and obey, for each u_{hk} and v_{hk} in $S_h^p(\Omega) \otimes P^q(\mathcal{I}^n)$,

$$(4.1) \qquad \int_{t^n}^{t^{n+1}} ((R_{hk})_t, u_{hk})dt + \int_{t^n}^{t^{n+1}} \left(\frac{2R_{hk}(H_{hk})_x}{\sqrt{2R_{hk} + (R_{hk})_x^2}}, (u_{hk})_x\right) dt = 0,$$

$$(4.2) \qquad \begin{aligned} \int_{t^n}^{t^{n+1}} (H_{hk}, v_{hk})dt &= \int_{t^n}^{t^{n+1}} \left(\frac{(R_{hk})_x}{\sqrt{2R_{hk} + (R_{hk})_x^2}}, (v_{hk})_x\right) dt \\ &\quad + \int_{t^n}^{t^{n+1}} \left(\frac{1}{\sqrt{2R_{hk} + (R_{hk})_x^2}}, v_{hk}\right) dt \end{aligned}$$

where R_{hk} at $t = t^n$ is fixed by continuity (or by the initial condition if $n = 0$).

The arguments which gave us (3.4) and (3.8) hold for the spaces to which R_{hk}, H_{hk}, u_{hk}, and v_{hk} belong for each finite element approximation, and each approximate solution (R_{hk}, H_{hk}) will show constancy of V and Φ and monotone decrease of Ψ.

In the simplest case, $p = 1$ and $q = 0$, we write $R_{hk}(x,t)$ on the time-strip $\mathcal{I}^n$ as

$$R_{hk}(x,t) = \frac{t - t^n}{k^n}[R_{hk}(x, t^{n+1}) - R_{hk}(x, t^n)] + R_{hk}(x, t^n),$$

and $H_{hk}(x,t)$ reduces to the constant-in-time function $H_{hk}(x,t) = H_{hk}(x, (t^n + t^{n+1})/2)$. Upon use of the usual expansion in piecewise linear basis functions, (4.1) and (4.2) yield a system of nonlinear algebraic equations for the coefficients of the basis functions. In our numerical work that algebraic system was solved by an iteration procedure which was initialized with the choice $R_{hk}^0(x, t^1) = R_{hk}(x, t^0)$ for the time-strip $\mathcal{I}^0 = [t^0, t^1]$ and carried forward with the extrapolation

$$R^0_{hk}(x,t^{n+1}) = \left(1+\frac{k^n}{k^{n-1}}\right)R_{hk}(x,t^n) - \frac{k^n}{k^{n-1}}R_{hk}(x,t^{n-1})$$

for the strip $\mathcal{I}^n$, $n \geq 1$; here R^l_{hk} denotes the lth iterate of R_{hk}. At the lth iteration on $\mathcal{I}^n$, $n \geq 0$, R^l_{hk}, and H^l_{hk} are determined by solving the linear system

$$\int_{t^n}^{t^{n+1}} ((R^l_{hk})_t, u_{hk})dt + \int_{t^n}^{t^{n+1}} \left(\frac{2R^{l-1}_{hk}(H^l_{hk})_x}{\sqrt{2R^{l-1}_{hk} + (R^{l-1}_{hk})^2_x}}, (u_{hk})_x\right)dt = 0,$$

$$\int_{t^n}^{t^{n+1}} (H^l_{hk}, v_{hk})dt = \int_{t^n}^{t^{n+1}} \left(\frac{(R^l_{hk})_x}{\sqrt{2R^{l-1}_{hk} + (R^{l-1}_{hk})^2_x}}, (v_{hk})_x\right)dt$$

$$+\int_{t^n}^{t^{n+1}} \left(\frac{1}{\sqrt{2R^{l-1}_{hk} + (R^{l-1}_{hk})^2_x}}, v_{hk}\right)dt$$

with u_{hk} and v_{hk} as in (4.1), (4.2) and

$$R^l_{hk}(x,t) = \frac{t-t^n}{k^n}[R^l_{hk}(x,t^{n+1}) - R_{hk}(x,t^n)] + R_{hk}(x,t^n).$$

5. Numerical experiments. For the two numerical experiments described below, we employed a finite element approximation of type (1, 0). The initial data had the form (1.6), (1.7) with each phase angle, φ_i, zero and, more importantly, each wave-number k_i greater than 1, so that (1.10) yields

$$\alpha(k_i) < 0, \qquad i = 1, \ldots, M, \tag{5.1}$$

and hence each exponential term in the solution (1.9) of the linear equation (1.8) decays to zero. The wave-numbers k_i were taken to be commensurate, which makes the periodic boundary conditions (2.12) exact relations.

Let $\bar{r}$ be given by

$$\pi(b-a)\bar{r}^2 = V(0) = \pi\int_a^b r_0^2 dx \tag{5.2}$$

and put

$$g(t) = \sup_x |r(x,t) - \bar{r}|. \tag{5.3}$$

A cylindrical body can be said to be *asymptotically stable against a class $\mathcal{P}$ of perturbations*, ϵu, if for each function $r_0 = 1 + \epsilon u$ with ϵu in $\mathcal{P}$ the solution of Problem RH obeys the two conditions

$$r(x,t) > 0 \qquad \forall(x,t) \in \Omega \times (0,\infty) \tag{I}$$

and

$$g(t) \to 0 \qquad \text{as} \qquad t \to \infty. \tag{II}$$

(I) asserts that "break-up" does not occur; (II) asserts that in the limit as $t \to \infty$ the body returns to cylindrical shape with a radius $\bar{r}$ determined by the mean volume $V(0)/(b-a)$ of the perturbed cylinder. (To generalize the condition (II) to initial data that are not periodic, but instead, say, almost periodic, one may replace $V(0)/(b-a)$ in the definition of $\bar{r}$ by the quantity

$$\langle \pi r^2 \rangle = 2\langle \pi R \rangle = \lim_{X\to\infty} \frac{\pi}{2X} \int_{-X+x_0}^{X+x_0} r^2 dx \tag{5.4}$$

which is constant in time and, for almost periodic functions, independent of x_0.)

When the function u in (1.6) is specified, the solution r of the nonlinear equation (1.5) depends on ϵ. For the perturbation analysis given in [2] we made the usual assumption that the dependence of $r(x, t; \epsilon)$ on ϵ is sufficiently smooth that for each integer n one can write

$$r(x, t; \epsilon) = 1 + \sum_{l=1}^{n} \epsilon^l w^{(l)}(x, t) + O(\epsilon^{n+1}). \tag{5.5}$$

By placing (5.5) in (1.5) and using (1.6) with u as in (1.7) we found, using a perturbation analysis of order 2, i.e., setting $n = 2$ in (5.5), that the term $1 + \epsilon w^{(1)}(x, t)$ is given by the right-hand side of (1.9), and

$$\begin{aligned} w^{(2)}(x, t) = {}& \frac{1}{4}\sum_{i=1}^{M} c_i^2 - \frac{1}{4}\sum_{i=1}^{M} c_i^2 e^{2\alpha(k_i)t} + \sum_{i=1}^{M} A_1(i; t)\cos(2k_i x + 2\varphi_i) \\ & + \sum_{1\le i<j\le M} A_2(i, j; t)\cos((k_i + k_j)x + \varphi_i + \varphi_j) \\ & + \sum_{1\le i<j\le M} A_3(i, j; t)\cos((k_i - k_j)x + \varphi_i - \varphi_j) \end{aligned} \tag{5.6}$$

where

$$A_1(i; t) = \frac{3c_i^2\beta(k_i)}{2[\alpha(2k_i) - 2\alpha(k_i)]}(e^{\alpha(2k_i)t} - e^{2\alpha(k_i)t}), \tag{5.7a}$$

$$A_2(i, j; t) = \frac{c_i c_j \beta^+(k_i, k_j)}{\alpha(k_i + k_j) - \alpha(k_i) - \alpha(k_j)}(e^{\alpha(k_i+k_j)t} - e^{(\alpha(k_i)+\alpha(k_j))t}), \tag{5.7b}$$

$$A_3(i, j; t) = \frac{c_i c_j \beta^-(k_i, k_j)}{\alpha(k_i - k_j) - \alpha(k_i) - \alpha(k_j)}(e^{\alpha(k_i-k_j)t} - e^{(\alpha(k_i)+\alpha(k_j))t}) \tag{5.7c}$$

with α as in (1.10) and

$$\beta(k) = k^2(1 + k^2), \tag{5.8a}$$

$$\beta^+(k_i, k_j) = (k_i k_j + k_i^2 + k_j^2)(1 + k_i k_j), \tag{5.8b}$$

$$\beta^-(k_i, k_j) = (k_i k_j - k_i^2 - k_j^2)(1 - k_i k_j). \tag{5.8c}$$

When, as in the numerical experiments, (1.11a) holds, the quantities $\alpha(k_i)$, $\alpha(2k_i)$, $\alpha(k_i + k_j)$ are negative for each i and $j > i$. Hence not only $w^{(1)}$ but also all the terms in $w^{(2)}$, other than the constant term $\frac{1}{4}\epsilon^2 \sum_{i=1}^{M} c_i^2$ and possibly some of the form $\epsilon^2 f(k_i, k_j; t)\cos((k_i - k_j)x + \varphi_i - \varphi_j)$ with

$$f(k_i, k_j; t) = \frac{c_i c_j \beta^-(k_i, k_j)}{\alpha(k_i - k_j) - \alpha(k_i) - \alpha(k_j)} e^{\alpha(k_i-k_j)t}, \tag{5.9}$$

will decay to zero exponentially with increasing t. The quantity $|f(k_i, k_j; t)|$ is constant in time when $|k_i - k_j| = 1$; it increases exponentially if and only if

$$0 < |k_i - k_j| < 1. \tag{5.10}$$

Thus, if (1.11a) holds and in addition $|k_i - k_j| > 1$ for all distinct pairs (i, j), the second-order perturbation analysis, like the linear theory, does not yield the existence of times at which $g(t)$ increases; however, a higher analysis may yield such times. In fact, exponential growth of $g(t)$, beginning at some time $t > 0$ and proceeding until break-up occurs, i.e., until a time t^* at which there are values x^* of x with $r(x^*, t^*) = 0$, will be shown in an nth-order perturbation analysis if (1.11b) holds for an M-tuple $(m_1, m_2, \ldots, m_M)$ with the m_i having (positive, negative, or zero) integral values.

In the numerical experiments reported here (1.11a) and (5.10) hold for at least one pair (i, j), $i < j$, and hence although the linear theory based on (1.8) predicts that the cylindrical body is asymptotically stable against the perturbation studied, the second-order theory predicts that the body is not stable against the perturbation and suggests that break-up can occur at a finite time. Specifically, under the conditions of the experiments, the second-order analysis yields the conclusion that for short times $r(x, t)$ is approximated by the expression

$$1 + \frac{\epsilon^2}{4}\sum_{i=1}^{M} c_i^2 + \epsilon \sum_{i=1}^{M} c_i e^{\alpha(k_i)t} \sin(k_i x + \varphi_i)$$

where each $\alpha(k_i)$ is negative, and hence, after a brief time interval, $r(x, t)$ will be close for all x to the constant

$$\bar{r} = \left[1 + \frac{\epsilon^2}{2}\sum_{i=1}^{M} c_i^2\right]^{1/2} = 1 + \frac{\epsilon^2}{4}\sum_{i=1}^{M} c_i^2 + O(\epsilon^4);$$

i.e., the body will be essentially indistinguishable from a cylinder. In a subsequent time interval, however, $r(x, t)$ will be approximated by

$$1 + \frac{\epsilon^2}{4}\sum_{i=1}^{M} c_i^2 + \epsilon^2 \sum_{(i,j)\in\Gamma} f(k_i, k_j; t)\cos((k_i - k_j)x + \varphi_i - \varphi_j)$$

where f is as in (5.9) and Γ is the set of pairs (i, j) with $1 \le i < j \le M$ and $|k_i - k_j| < 1$; during that interval $g(t)$ will increase monotonically.

Let ν be the minimum value that $g(t)$ must have for a departure of the body from cylindrical shape to be easily observable after $g(t)$ has decayed and started to increase. The time $t_\#$ at which $g(t)$ attains the value ν (after an initial decrease) may be called the "incubation time" for observable growth of a perturbation which according to the linear theory (1.8) would only decay. The second-order perturbation analysis gives the following relation for $t_\#$ in the case in which there is precisely one pair (i, j) for which (5.10) holds:

$$\nu = \epsilon^2 f(k_i, k_j; t_\#). \tag{5.11}$$

A reasonable value for ν would be 0.05. We note that equation (5.11) implies that $t_\#$ varies slowly, i.e., logarithmically, with ϵ.

For large k, $\alpha(k)$ decreases rapidly with k; indeed, as $\alpha(k) \sim -k^4$. However, on the interval $0 < k < 1$ where $\alpha(k)$ is positive, the maximum value of $\alpha(k)$, $\alpha(1/\sqrt{2})$, is only $\frac{1}{4}$. Hence the incubation time $t_\#$ can be expected to be orders of magnitude longer than the time required for decay of an initial perturbation that has each k_i appreciably greater than 1. This disparity in time scales requires that time steps be adjusted in numerical calculations in accord with the rate of convergence of the iterative procedure described at the end of §4, i.e., in accord with the rate of evolution of R and H.

For the first numerical experiment the initial data correspond to the function

$$r_0(x) = 1 + 5 \times 10^{-2} [\sin(5x) + \sin(11x/2)], \tag{5.12}$$

which has minimum period 4π. Here $k_1 = 5 > 1$, $k_2 = 11/2 > 1$, but $|k_1 - k_2| = 1/2 < 1$. According to the second-order perturbation analysis, the two sinusoidal terms in the initial data should decay rapidly, and, after this rapid decay, which in the present case lasts until approximately $t = 2 \times 10^{-2}$, $r(x, t)$ should be, for a while, close to a cosine function of x which has period 4π and an amplitude growing as $e^{\alpha(1/2)t}$ with $\alpha(1/2) = 3/16$. In the finite element scheme, the interval $a \le x \le b$ was chosen to have length 8π, i.e., two periods, and was discretized into 512 equal segments. For the time interval $0 < t \le 5 \times 10^{-2}$ that contains the times of rapid decay of $g(t)$, the time step, Δt, was chosen to be 10^{-4}; during the early growth phase, i.e., for $5 \times 10^{-2} < t \le 28.05$, Δt was set equal to $t = 10^{-2}$. For $28.05 \le t \le 28.27234 = t^*$, the time steps were refined in such a way that Δt decreased rapidly as t approached the break-up time t^*. (As is common practice, we repeated suspected critical parts of this and the second numerical experiment using refined spatial and temporal meshes; the reported mesh densities are such that further refinement produced no change in results.)

In Fig. 1 there are graphs of r versus x for various t between 0 and t^*. The initial data, (5.12), are shown in Fig. 1a. The scale of the ordinate r contracts in the sequence from Fig. 1b to Fig. 1f, but in each case is greater than that of the abscissa x. Figure 1b contains graphs corresponding to $t = 10^{-2}$ and $t = 1.63 \times 10^{-2}$. At $t = 10^{-2}$ remnants of the terms $5 \times 10^{-2} \sin(5x)$ and $5 \times 10^{-2} \sin(11x/2)$ can be seen perturbing the function $\tilde{r}(\cdot, t)$ given by

$$\tilde{r}(x, t) = \bar{r} + \epsilon^2 f(5, 11/2; t) \cos(x/2) \tag{5.13}$$

where $\epsilon^2 = 25 \times 10^{-4}$ and f is as in (5.9) with $\alpha = 3/16$. At $t = 1.63 \times 10^{-2}$, $g(t)$ attains its minimum value, and at that time the difference between $r(x, t)$ and $\tilde{r}(x, t)$ is not detectable on the scale employed for r of Fig. 1b. Figure 1c contains graphs for values of t at which $r(x, t)$ is close to $\tilde{r}(x, t)$; the earliest time at which the second-order perturbation analysis gives results (here plotted with dashes) that are distinguishable from the finite element results (on the scale of Fig. 1c) is $t = 15$.

In Fig. 2, where we employ equal scales for the ordinate and the abscissa, there are shown profiles of the axially symmetric body whose surface is given by $r = r(x, t)$. The initial configuration is seen in Fig. 2a. At a time $t \sim 10^{-2}$ the body is of cylindrical shape to within an error of the order 0.1% in r. If we set $\nu = 0.05$, the configuration shown in Fig. 2b, i.e., that for $t = 20$, is very close to one with $g(t) = \nu$; in fact, the finite element results yield 0.05 for $g(t)$ when $t = 19.55$; the second-order perturbation analysis, on the other hand, yields 0.05 for $g(t)$ when $t = 19.72$. We think it remarkably fortunate that in this case an elementary analysis that takes into account only the lowest-order nonlinear terms gives an estimate for the incubation time $t_\#$ that is off by only 1%.

There are cases in which the evolution of the subbodies formed at time t^* can be studied for $t > t^*$ by reparameterizing with spherical coordinates the surface of each connected subbody. In this procedure one chooses a point on the original x-axis to be the origin of a spherical coordinate system, and the closed surface of the subbody containing that origin is described by giving the distance $\hat{r}$ from the origin to a point on the surface as a function of t and the colatitude φ measured from the x-axis. A finite element method analogous to

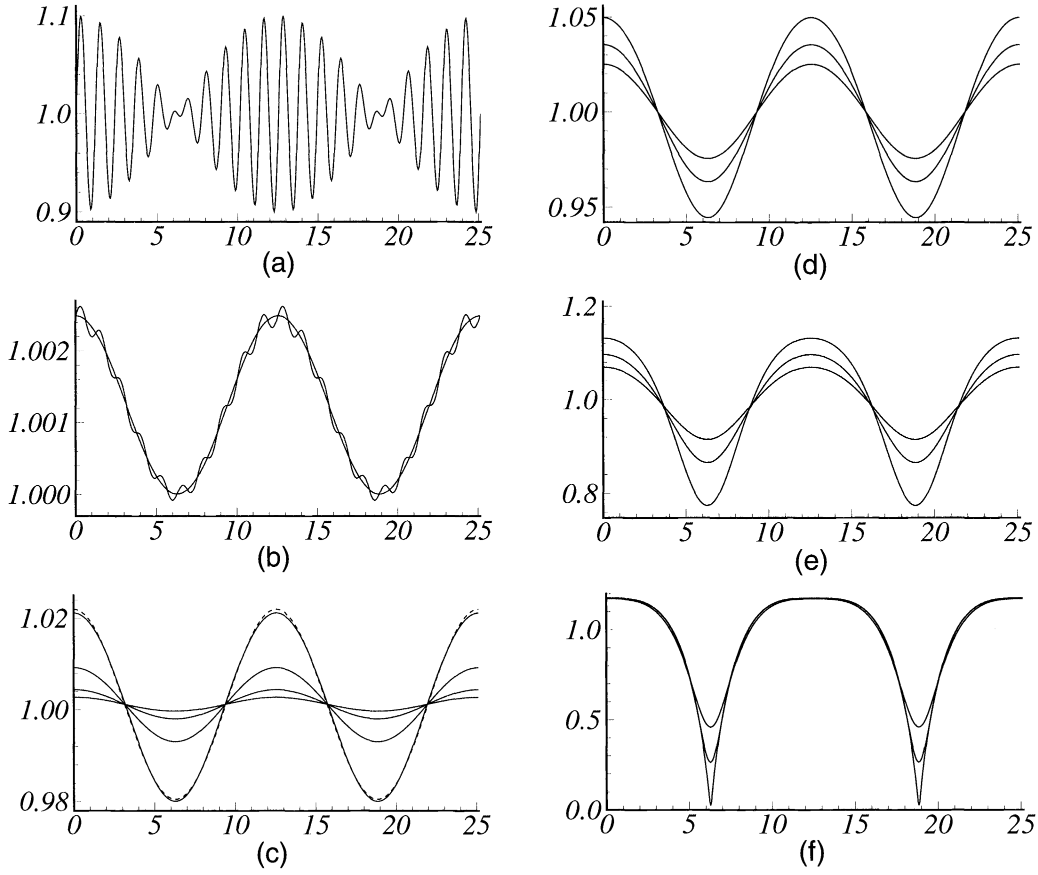


FIG. 1. *The radius r as a function of x at various times t in the first numerical experiment*: (a) $t = 0$; (b) $t = 10^{-2}$ (*wavy line*), $t = 1.63 \times 10^{-2}$; (c) $t = 0.1, 5, 10, 15$ (*the dashed curve gives the result of the second-order perturbation analysis at* $t = 15$); (d) $t = 16, 18, 20$; (e) $t = 22, 24, 26$; (f) $t = 28, 28.25, 28.27234$.

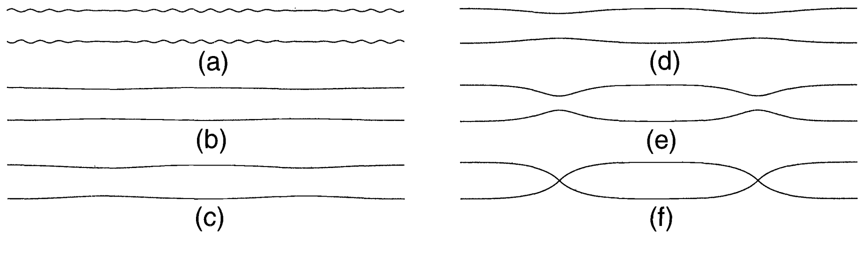


FIG. 2. *Profiles at selected times in the first experiment*: (a) $t = 0$; (b) $t = 20$; (c) $t = 23$; (d) $t = 25.5$; (e) $t = 28$; (f) $t = t^* = 28.27234$.

that employed for the cylindrical-coordinate formulation gives satisfactory results in cases in which, for *all* $t > t^*$, the closed surface remains star shaped with respect to a fixed point. We note that for motion by Laplacian of mean curvature there is no analogue of Huisken's theorem [7] in the theory of motion by mean curvature, asserting that if a closed surface deforming in accord with (1.4) is convex at one time t^0 then it is convex at all subsequent times.

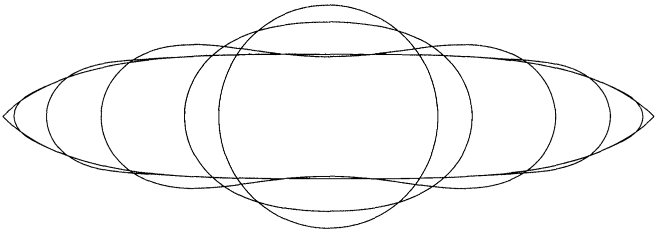


FIG. 3. *Profiles in the first experiment at times* $t = t^* + \hat{t}$ *for* $\hat{t} = 0, 10^{-3}, 10^{-1}, 1, 4, 10$. *Note that at* $\hat{t} = 0, 10^{-3}$, *and* 10^{-1} *the subbody is convex and that this convexity, which is not present at* $\hat{t} = 1$, *returns before* $\hat{t} = 4$.

In the experiment under discussion here, the subbodies are congruent and for all $t > t^*$ are star shaped with respect to the midpoints of successive values of x^*. Calculated profiles of a subbody for several times $t = t^* + \hat{t}$ with $\hat{t} > 0$ are seen in Fig. 3. When $\hat{t} = 10$ each component is a sphere in the sense that $\hat{r}(\varphi, t^* + 10)$ is constant in φ to within six significant figures; of course, the collection of spheres so obtained is the new equilibrium state of the original perturbed cylinder.

The initial data for the second numerical experiment correspond to

$$\begin{aligned} r_0(x) = 1 + 10^{-2}[&\sin(2x) + \sin(13x/6) + \sin(7x/3) \\ &+ \sin(5x/2) + \sin(8x/3) + \sin(17x/6)]. \end{aligned} \tag{5.14}$$

In this case r_0 has minimum period 12π. For the finite element computation, $b - a$ was chosen to be 12π, i.e., one period. As in the previous case, the interval $a \le x \le b$ was discretized into 512 equal segments, and the time steps Δt were chosen to be 10^{-4} on an interval that contained the times of initial decay, 10^{-2} during the early growth phase and much smaller as t approached t^*. Numerical results are shown in Figs. 4 and 5. Here, again, albeit only wave-numbers k_i that exceed 1 are present in the initial perturbation, and hence the linear theory predicts that $g(t)$ should decay to zero exponentially in t. A second-order perturbation analysis yields the conclusion that new wave-numbers given by $|k_i - k_j|$ will appear and grow. In the present case there are five distinct values of $|k_i - k_j|$ obeying (5.10), namely, 1/6, 1/3, 1/2, 2/3, 5/6. Of these, three, 1/2, 2/3, 5/6, have rates of growth, $\alpha(1/2) = 3/16 = 0.1875$, $\alpha(2/3) = 20/9^2 \simeq 0.2469$, $\alpha(5/6) = 275/36^2 \simeq 0.2122$, that are close to maximum growth rate $\alpha(1/\sqrt{2}) = 0.25$. The periods $2\pi|k_i - k_j|^{-1}$ corresponding to these three fast-growing modes are 4π, 3π, $12\pi/5$. The distances between adjacent local minima of $r(x, t^*)$ were found to be 9.35115, 9.42478, and 9.57204. One of these values is equal, up to five decimal places, to $3\pi \simeq 9.42478$, and the other two are much nearer to 3π than to either 4π or $12\pi/5$. As $\alpha(1/2)$ and $\alpha(5/6)$ both are less than $\alpha(2/3)$, it is not surprising that an examination of spatial frequencies at $t = t^*$ confirms that the mode with wave-number 2/3 (period 3π) dominates the growth.

Whereas in the first experiment the body breaks into congruent subbodies, in this experiment the break-up at time t^* yields two distinct equivalence classes $\mathcal{C}_1$ and $\mathcal{C}_2$ of congruent bodies. Figure 5d makes this clear. At $t = t^*$, the bodies in $\mathcal{C}_1$ have length 9.57204, and those in $\mathcal{C}_2$ have length $28.12708 = 2 \times 9.33115 + 9.42478$. We have found that members of $\mathcal{C}_1$ remain star shaped for all $t > t^*$, and the spherical-coordinate formulation can be employed to follow their evolution to a final equilibrium state which is spherical. On the other hand, the bodies in $\mathcal{C}_2$ cease to be star shaped at a finite time, and for them the spherical-coordinate formulation is not applicable without major modification. Here the question arises as to whether subbodies of class $\mathcal{C}_2$ will exhibit further break-up or end up as a single sphere. The search for a general algorithm to study post break-up behavior is a topic of current research.

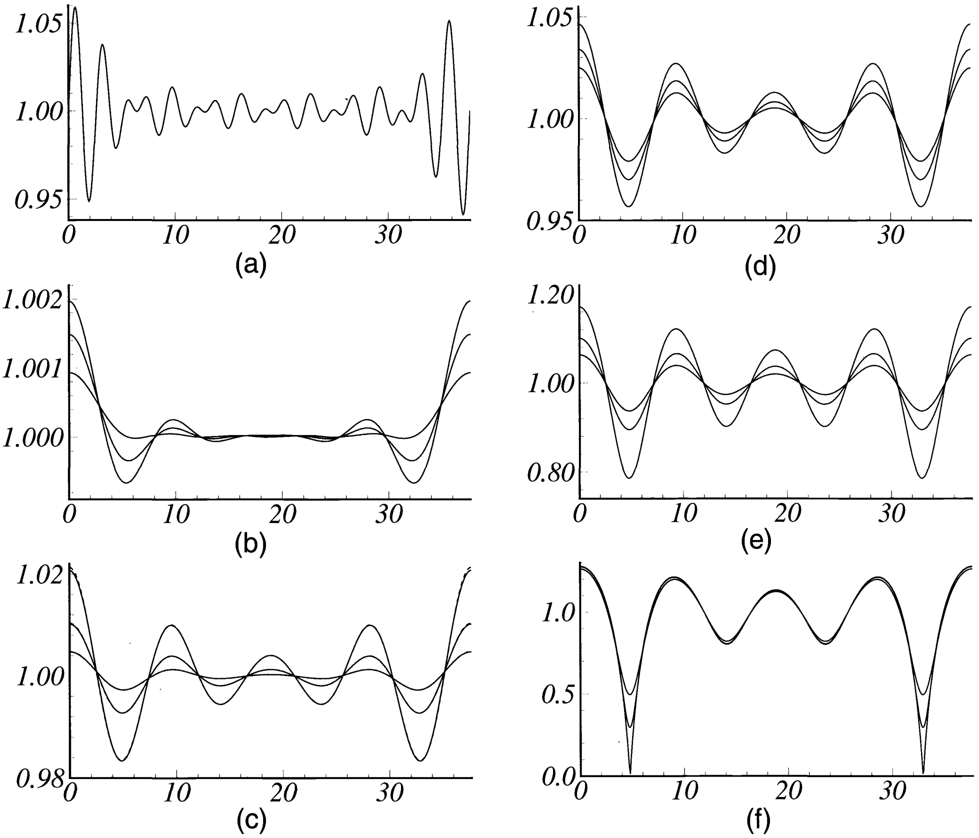


FIG. 4. *The radius r as a function of x in the second numerical experiment*: (a) $t = 0$; (b) $t = 0.7, 4.7, 6.7$; (c) $t = 12, 16, 19$; (d) $t = 20, 22, 24$; (e) $t = 26, 28, 30$; (f) $t = 31.5, 31.7, 31.7357$.

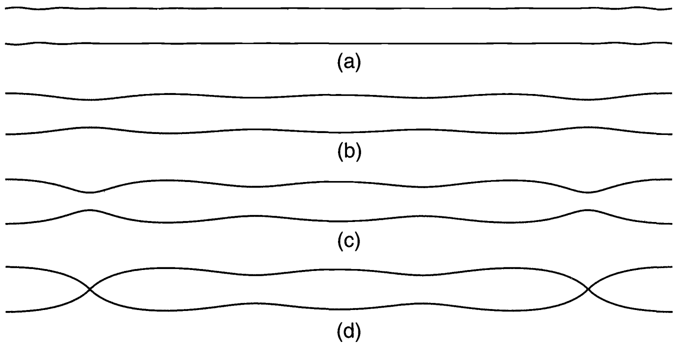


FIG. 5. *Profiles in the second experiment*: (a) $t = 0$; (b) $t = 30$; (c) $t = 31.5$; (d) $t = t^* = 31.7357$.

REFERENCES

[1] J. W. CAHN AND J. E. TAYLOR, *Surface motion by surface diffusion*, Acta Metall. Mater., 42 (1994), pp. 1045–1063.

[2] B. D. COLEMAN, R. S. FALK, AND M. MOAKHER, *Stability of cylindrical bodies in the theory of surface diffusion*, Phy. D, 89 (1995), pp. 123–135.

[3] G. DZIUK, *An algorithm for evolutionary surfaces*, Numer. Math., 58 (1991), pp. 603–611.

[4] C. M. Elliott and H. Garcke, *Existence results for diffusive surface motion laws*, preprint.

[5] L. C. Evans and J. Spruck, *Motion of level sets by mean curvature* I, J. Differential Geom., 33 (1991), pp. 635–681.

[6] C. Herring, *Surface diffusion as a motivation for sintering*, in The Physics of Powder Metallurgy, W. E. Kingston, ed., McGraw Hill, New York, 1951, pp. 143–179.

[7] G. Huisken, *Flow by mean curvature of convex surfaces into spheres*, J. Differential Geom., 20 (1984), pp. 237–266.

[8] ———, *Local and global behaviour of hypersurfaces moving by mean curvature*, in Differential Geometry: Partial Differential Equations on Manifolds, R. Greene and S. T. Yau, eds., American Mathematical Society, Providence, RI, 1993, pp. 175–191.

[9] W. W. Mullins, *Theory of thermal grooving*, J. Appl. Phys., 28 (1957), pp. 333–339.

[10] F. A. Nichols and W. W. Mullins, *Surface- (interface-) and volume-diffusion contribution to morphological changes driven by capillarity*, Trans. Metall. Soc., A.I.M.E., 233 (1965), pp. 1840–1847.

[11] J. A. Sethian, *Curvature and the evolution of fronts*, Comm. Math. Phys., 101 (1985), pp. 487–499.

[12] H. M. Soner and P. E. Songanidis, *Singularities and uniqueness of cylindrically symmetric surfaces moving by mean curvature*, Comm. Partial Differential Equations, 18 (1993), pp. 859–894.

SIAM J. SCI. COMPUT.
Vol. 17, No. 6, pp. 1449–1474, November 1996

009

A FAST MULTIGRID ALGORITHM FOR ISOTROPIC TRANSPORT PROBLEMS II: WITH ABSORPTION*

T. MANTEUFFEL†, S. McCORMICK†, J. MOREL‡, AND G. YANG§

Abstract. A multigrid method for solving the one-dimensional slab-geometry S_N equations with isotropic scattering and absorption is presented. The case with no absorption was treated in part I of this paper [Manteuffel, McCormick, Morel, Oliveira, and Yang, *SIAM J. Sci. Comput.*, 16 (1995), pp. 601–635]. Relaxation is based on a two-cell inversion, which is very efficient because it takes advantage of the structure of the two-cell problem. For interpolation we use kinked linear elements. The kink is based on the amount of absorption present. The restriction operator is full weighting. Numerical results show this algorithm to be faster than diffusion synthetic acceleration (DSA) in all regimes. This scheme is also well suited for massively parallel computer architectures.

Key words. multigrid, particle transport

AMS subject classifications. 65N20, 65F10

1. Introduction.

In this paper we describe a fast method for solving the equations used to model the transport of neutral particles with isotropic scattering in slab geometry. These problems are important in many applications such as nuclear reactor design, radiation therapy in medical science, and radiation effects on global weather, and they are a fundamental part of many algorithms used to model more complicated applications such as coupled photon/electron scattering used to model satellite electronics shielding.

We focus on the solution of the steady-state, monoenergetic, linear Boltzmann equation in slab geometry. The need to solve such equations arises from the time-dependent Boltzmann equations used in the applications mentioned above. The physical domain is assumed to be a semi-infinite slab of width $b - a$ in the x dimension. Although they are three dimensional, we assume that the flux of particles is independent of the y and z coordinates. Let $\psi(x, \mu, e, t)$ represent the flux of particles at a position x, traveling at an angle μ to the x-axis, with energy e, at time t. The Boltzmann equation takes the form

$$\frac{\partial \psi}{\partial t} = -\mu \frac{\partial \psi}{\partial x} - \sigma_t(x, e)\psi + \int\int K(\mu' \to \mu, e' \to e)\psi(x, \mu', e', t)d\mu' de' + q(x, \mu, e, t). \tag{1.1}$$

Here, σ_t is the collision cross section and $\int_{x_0}^{x_1} \sigma_t dx$ represents the expected number of collisions that a particle will experience in the interval (x_0, x_1). In the same manner, the scattering kernel $\int\int K(\mu' \to \mu, e' \to e)$ represents the expected number of particles that will scatter from direction μ' and energy e' to direction μ and energy e. Finally, $q(x, \mu, e, t)$ represents particle sources.

The above equation is usually integrated with implicit time-stepping methods and the energy variable is treated with a piecewise-constant finite element method to yield a coupled system of equations at each time step. Let $\psi^k(x, \mu)$ be the flux of particles with energy e_k. Then

$$\mu \frac{\partial \psi^k}{\partial x} + \sigma_t^k(x)\psi^k = \sum_j \int K_{k,j}(\mu' \to \mu)\psi^j(x, \mu')d\mu' + q^k(x, \mu), \tag{1.2}$$

*Received by the editors July 1, 1993; accepted for publication (in revised form) June 19, 1995. This work was supported by AFOSR grant AFOSR 91-0156, the NSF grant DMS-8704169, the DOE grant DE-FG03-93ER25165 and Los Alamos National Laboratory.

†Program in Applied Mathematics, University of Colorado at Boulder, Boulder, CO 80309-0526 (tmanteuf@boulder.colorado.edu, stevem@boulder.colorado.edu).

‡Computer Research Group (C-3), Los Alamos National Laboratory, Los Alamos, NM 87545 (jim@c3.lanl.gov).

§Center for Computational Mathematics, University of Colorado at Denver, Denver, CO 80217-3364.

where the time-step information has been incorporated into σ_t^k and q^k, and $\int K_{k,j}(\mu' \to \mu)$ represents the expected number of collisions that will rescatter a particle from energy e_j into energy e_k.

Since particles generally lose energy, the system of equations is solved by a block Gauss–Seidel-type iteration. Starting with the highest energy level, $\psi^k(x, \mu)$ is updated assuming that $\psi^j(x, \mu)$ for $j \neq k$ are known. This yields the equation

$$\mu \frac{\partial \psi^k}{\partial x} + \sigma_t^k(x)\psi^k = \int K_{k,k}(\mu' \to \mu)\psi^k(x, \mu')d\mu' + \hat{q}^k(x, \mu), \tag{1.3}$$

where now $\hat{q}^K$ contains all the inscatter from other energy levels.

The kernel $K_{k,k}$ is often mildly anisotropic, becoming more isotropic as energy decreases. In this paper we focus on the isotropic form of (1.3) (for strongly anisotropic scattering see [14]):

$$\mu \frac{\partial \psi}{\partial x} + \sigma_t \psi = \frac{\sigma_s}{2} \int_{-1}^{1} \psi(x, \mu')d\mu' + q(x, \mu). \tag{1.4}$$

Here, σ_s is now the scattering cross section and represents the expected number of collisions that result in a rescatter, while $\sigma_a = \sigma_t - \sigma_s$ represents the expected number of collisions that result in an absorption. When $\sigma_t = \sigma_s$, there is no absorption. The equation is well defined if the flux of particles entering the slab is given as boundary conditions

$$\psi(a, \mu) = g_a(\mu), \quad \psi(b, -\mu) = g_b(\mu) \quad \mu\epsilon(0, 1). \tag{1.5}$$

In general, numerical methods that are effective for isotropic equations also work well for mildly anisotropic equations. We believe that this will be especially true for the multigrid algorithm described below because of the special form of relaxation used (see §3).

Problem (1.4) will inherit the discretization scheme from problem (1.1). The standard approach is to expand the angular dependence in terms of the first N Legendre polynomials and to close the system with a Galerkin condition. This is attractive because the Legendre polynomials are the eigenvectors of the scattering operators $K_{k,j}$ in slab geometry. This results in the P_{N-1} discretization. In slab geometry, this is equivalent to collocating at Gauss quadrature points which are known as the discrete ordinates or S_N discretization (cf. [7, 13]).

Spatial discretization must take into account the important optically dense or thick limit in which the problem becomes ill conditioned. Physically, this corresponds to a mean-free path between collisions that is small compared with the width of the slab and materials that allow very little absorption. Mathematically, we have

$$\sigma_t \to \infty, \quad \frac{\sigma_s}{\sigma_t} \to 1. \tag{1.6}$$

Dividing (1.1) by σ_t and taking the limits in (1.4) yields

$$\psi(x, \mu) = \frac{1}{2} \int_{-1}^{1} \psi(x, \mu')d\mu', \tag{1.7}$$

which admits any $\psi(x, \mu)$ that is independent of μ. Thus, in this limit, (1.4) is singularly perturbed with a large near null space. For discrete equations it is the product $\sigma_t h_i$, where h_i is a mesh parameter, that parameterizes each equation. There are two quantities that determine the character of the discrete problem: the overall thickness $\sigma_t(b - a)$ and the thickness of individual cells $\sigma_t h_i$.

Some spatial discretization schemes, for example upwind differences, have discretization error $O(\sigma_t h_i)$. Thus, for $\sigma_t \gg 1$ these schemes require $h_i \to 0$ even for well-behaved solutions. For a discussion of this issue see [4, 10] and Larsen and Morel [6]. One difference scheme that behaves well in the thick limit is the modified linear discontinuous (MLD) scheme [6]. Not only does it give the proper behavior in the thick limit, but it is very accurate. In [10], we developed a very efficient multigrid algorithm to solve the discrete equations produced by the MLD scheme in the absence of absorption ($\sigma_t = \sigma_s$). Our analysis shows that, in this case, a $V(1,1)$ cycle version of our algorithm yields a convergence factor ρ bounded as follows:

(a) for $\max_i(\sigma_t h_i) \ll 1, \quad \rho = O(\max_i(\sigma_t h_i)^2)$,

(b) for $\min_i(\sigma_t h_i) \gg 1, \quad \rho = O(\max_i(\frac{1}{\sigma_t h_i}))$.

Numerical results are even better than these theoretical estimates. Experiments yield the following convergence factor ρ for a $V(1,1)$ cycle:

(a) for $\max_i(\sigma_t h_i) \ll 1, \quad \rho = O(\max_i(\sigma_t h_i)^3)$,

(b) for $\min_i(\sigma_t h_i) \gg 1, \quad \rho = O(\frac{1}{\min_i(\sigma_t h_i)^2})$.

When there is absorption in the transport equations ($\sigma_t > \sigma_s$) and $\sigma_t h \gg 1$, the two-cell μ-line relaxation used in [10] will leave an error that is essentially independent of angle and continuous but no longer linear across two cells (see [10, §3]). The deviation of the errors from linearity depends on the value of $\gamma = \frac{\sigma_s}{\sigma_t}$ and h. Because the errors after relaxation will not be linear across two-cell pairs, the linear interpolation used in our previous multigrid algorithm is unsuitable. In this paper, we will present a new method which uses kinked elements whose shape will be determined by the severity of deviation from linearity of the errors after relaxation. When $\gamma = 1$, the kinked element will be linear.

We remark that for small values of the angular discretization dimension N the discrete form of (1.4) could easily be solved by direct inversion of the resulting banded matrix. However, direct methods require $O(N^2m)$ operations and $O(N^2m)$ storage, where m is the number of spatial cells. Both the method described here and the diffusion synthetic acceleration (DSA) [1, 2, 5] to which we compare our method require only $O(Nm)$ operations and storage. Thus, these methods are viable for problems that require fine angular resolution. Because of the large number of times equations of the form (1.4) must be solved, efficiency is critical. Moreover, both DSA and our method have obvious generalization to higher spatial dimensions.

In §2, we analyze the properties of relaxation when $\gamma \neq 1$. In §3, we introduce the kinked element, relaxation-induced interpolation, and coarse grid operators used to solve the transport problems when $\gamma \neq 1$. The multigrid algorithm is discussed in §4. In §5, we examine the properties of the new interpolation. In §6, computational results are presented. We show that the multigrid algorithm is faster than the DSA algorithm [1, 2, 5] on a representative set of test problems. We include a study of domains with nonhomogeneous material properties. Finally, in §7, we present our conclusions.

2. The discrete model. We start with an S_N approximation in angle. This corresponds to expanding the angle dependence in terms of the first N Legendre polynomials. Closing the equations with a Galerkin formulation yields a system that resembles collocation at the Gauss quadrature points $\mu_j, j = 1, \ldots, N$. Similarly, the integral is replaced with a quadrature formula using the Gauss quadrature weights $\omega_j, j = 1, \ldots, N$. By symmetry, we may denote the positive quadrature points $\mu_j > 0, j = 1, \ldots, n$ with corresponding weights $\omega_j, j > 1, \ldots, n$, and negative quadrature points $-\mu_j < 0, j = 1, \ldots, n$ with corresponding weights $\omega_j, j = 1, \ldots, n$, where $n = N/2$. We use $\psi^+(x_i, \mu_j) = \psi(x_i, \mu_j)$, $\psi^-(x_i, \mu_j) = \psi(x_i, -\mu_j)$ to denote flux variables at spatial point x_i, angular position μ_j at positive and negative directions, respectively.

The spatial difference scheme is the MLD discretization. Consider the grid $a = x_{1/2} < x_{3/2} < \cdots < x_{m+1/2} = b$, where $x_{i\pm 1/2}$ are cell edges and $x_i = (x_{i-1/2} + x_{i+1/2})/2$ is the cell center. Let $\underline{\psi}_i^+ = (\psi^+(x_i, \mu_1), \ldots, \psi^+(x_i, \mu_n))^T$ and $\underline{\psi}_i^- = (\psi^-(x_i, \mu_1), \ldots, \psi^-(x_i, \mu_n))^T$.

In matrix form MLD can be written as (for a complete presentation see [6], [10])

$$B_i(\underline{\psi}^+_{i+\frac{1}{2}} - \underline{\psi}^+_{i-\frac{1}{2}}) + \underline{\psi}^+_i = \gamma R(\underline{\psi}^+_i + \underline{\psi}^-_i) + \underline{q}^+_i, \tag{2.1a}$$

$$2B_i(\underline{\psi}^+_{i+\frac{1}{2}} - \underline{\psi}^+_i) + \underline{\psi}^+_{i+\frac{1}{2}} = \gamma R(\underline{\psi}^+_{i+\frac{1}{2}} + 2\underline{\psi}^-_i - \underline{\psi}^-_{i-\frac{1}{2}}) + \underline{q}^+_{i+\frac{1}{2}} \tag{2.1b}$$

for $i = 1, \ldots, m$ and

$$B_i(\underline{\psi}^-_{i-\frac{1}{2}} - \underline{\psi}^-_{i+\frac{1}{2}}) + \underline{\psi}^-_i = \gamma R(\underline{\psi}^+_i + \underline{\psi}^-_i) + \underline{q}^-_i, \tag{2.1c}$$

$$2B_i(\underline{\psi}^-_{i-\frac{1}{2}} - \underline{\psi}^-_i) + \underline{\psi}^-_{i-\frac{1}{2}} = \gamma R(\underline{\psi}^-_{i-\frac{1}{2}} + 2\underline{\psi}^+_i - \underline{\psi}^+_{i+\frac{1}{2}}) + \underline{q}^-_{i-\frac{1}{2}} \tag{2.1d}$$

for $i = 1, \ldots, m$ with boundary conditions

$$\underline{\psi}^+_{\frac{1}{2}} = \underline{g}^+_a, \quad \underline{\psi}^-_{m+\frac{1}{2}} = \underline{g}^-_b, \tag{2.1e}$$

where B_i and R are the $n \times n$ matrices

$$B_i = \begin{bmatrix} \frac{\mu_1}{\sigma_t h_i} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{\mu_n}{\sigma_t h_i} \end{bmatrix}, \quad R = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \begin{bmatrix} \omega_1 & \cdots & \omega_n \end{bmatrix} = \underline{1\omega}^T. \tag{2.1f}$$

In block matrix form, (2.1) can be written as

(2.2)
$$\begin{bmatrix} I + 2B_i - \gamma R & -2\gamma R & -2B_i & \gamma R \\ 0 & I - \gamma R & -\gamma R & B_i \\ B_i & -\gamma R & I - \gamma R & 0 \\ \gamma R & -2B_i & -2\gamma R & I + 2B_i - \gamma R \end{bmatrix} \begin{bmatrix} \underline{\psi}^-_{i-\frac{1}{2}} \\ \underline{\psi}^+_i \\ \underline{\psi}^-_i \\ \underline{\psi}^+_{i+\frac{1}{2}} \end{bmatrix} = \begin{bmatrix} \underline{q}^-_{i-\frac{1}{2}} \\ B_i\underline{\psi}^+_{i-\frac{1}{2}} + \underline{q}^+_i \\ B_i\underline{\psi}^-_{i+\frac{1}{2}} + \underline{q}^-_i \\ \underline{q}^+_{i+\frac{1}{2}} \end{bmatrix}.$$

In this paper, we will use an edge–edge notation for the unknown flux variable $\underline{\psi}$. That is, the flux within cell i is linear and is determined by the value at the right and left sides $\underline{\psi}^+_{ir}$ and $\underline{\psi}^+_{il}$. We define the transformation

$$\begin{bmatrix} \underline{\psi}^-_{il} \\ \underline{\psi}^+_{il} \\ \underline{\psi}^-_{ir} \\ \underline{\psi}^+_{ir} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & -1 \\ -1 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \underline{\psi}^-_{i-\frac{1}{2}} \\ \underline{\psi}^+_i \\ \underline{\psi}^-_i \\ \underline{\psi}^+_{i+\frac{1}{2}} \end{bmatrix}. \tag{2.3a}$$

Then the inverse transformation of (2.3a) will be

$$\begin{bmatrix} \underline{\psi}^-_{i-\frac{1}{2}} \\ \underline{\psi}^+_i \\ \underline{\psi}^-_i \\ \underline{\psi}^+_{i+\frac{1}{2}} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \underline{\psi}^-_{il} \\ \underline{\psi}^+_{il} \\ \underline{\psi}^-_{ir} \\ \underline{\psi}^+_{ir} \end{bmatrix}. \tag{2.3b}$$

Figure 2.1 shows the piecewise discontinuous elements we use. At cell i, two variables $\underline{\psi}_{il}$, $\underline{\psi}_{ir}$ are defined.

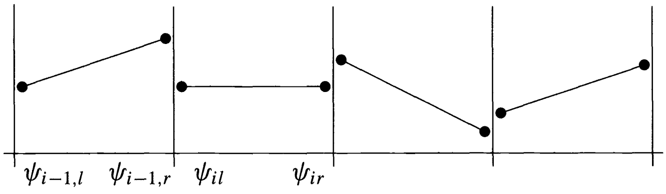


FIG. 2.1. *Piecewise discontinuous elements.*

By substituting (2.3b) into (2.2) and multiplying both sides of (2.2) by the transformation matrix of (2.3a), we then can write (2.2) in block matrix form as

$$(2.4)\qquad \begin{bmatrix} A_1 & -C_1 & & & & \\ -D_2 & A_2 & -C_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & -D_i & A_i & -C_i & \\ & & & \ddots & \ddots & \ddots \\ & & & & -D_m & A_m \end{bmatrix} \begin{bmatrix} \underline{\Psi}_1 \\ \underline{\Psi}_2 \\ \vdots \\ \underline{\Psi}_i \\ \vdots \\ \underline{\Psi}_m \end{bmatrix} = [\ \underline{Q}\ ],$$

where $\underline{Q}$ is the right-hand side and

$$(2.5a)\qquad A_i = \begin{bmatrix} I + B_i - \gamma R & -\gamma R & -B_i & 0 \\ -\gamma R & I + B_i - \gamma R & 0 & B_i \\ B_i & 0 & I + B_i - \gamma R & -\gamma R \\ 0 & -B_i & -\gamma R & I + B_i - \gamma R \end{bmatrix},$$

$i = 1, \dots, m,$

$$(2.5b)\qquad C_i = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 2B_i & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad i = 1, \dots, m-1,$$

$$(2.5c)\qquad D_i = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2B_i \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad i = 2, \dots, m,$$

$$(2.5d)\qquad \underline{\Psi}_i = (\underline{\psi}_{il}^-, \underline{\psi}_{il}^+, \underline{\psi}_{ir}^-, \underline{\psi}_{ir}^+)^T, \quad i = 1, \dots, m.$$

In the following section, we will examine the two-cell μ-line relaxation and its properties.

3. Relaxation. By two-cell red-black μ-line relaxation, we mean that at cells $2i - 1$ and $2i$, we set $\underline{\psi}^+_{(2i-2)r}$ and $\underline{\psi}^-_{(2i+1)l}$ as boundary values and solve for all other interior values in cells $2i - 1$ and $2i$ simultaneously. This relaxation is carried out in a red-black ordering. Since relaxation uses a red-black ordering, we can look at each two-cell pair individually. For a two-cell pair, for example cells $2i - 1$ and $2i$, the errors after the relaxation will be

$$\begin{bmatrix} \underline{e}^-_{(2i-1)l} \\ \underline{e}^+_{(2i-1)l} \\ \underline{e}^-_{(2i-1)r} \\ \underline{e}^+_{(2i-1)r} \\ \underline{e}^-_{(2i)l} \\ \underline{e}^+_{(2i)l} \\ \underline{e}^-_{(2i)r} \\ \underline{e}^+_{(2i)r} \end{bmatrix} = \begin{bmatrix} A_{2i-1} & -C_{2i-1} \\ -D_{2i} & A_{2i} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 2B_{2i-1}\hat{\underline{e}}^+_{(2i-2)r} \\ 0 \\ 0 \\ 0 \\ 0 \\ 2B_{2i}\hat{\underline{e}}^-_{(2i+1)l} \\ 0 \end{bmatrix}, \tag{3.1}$$

where $\hat{\underline{e}}$ on the right-hand side is the error at the outside border of the two-cell pair before relaxation. As we have shown in our previous paper [10], the inversion of the

$$\begin{bmatrix} A_{2i-1} & -C_{2i-1} \\ -D_{2i} & A_{2i} \end{bmatrix}$$

can be performed in $O(n)$ operations by using the Sherman–Morrison formula. In [10], only the case $\gamma = 1$ was examined. The following result for $\gamma = 1$ was proved.

THEOREM 1. *Suppose* $\gamma = 1$ *in* (3.1) *and* $\min(\sigma_t h_{2i-1}, \sigma_t h_{2i}) \gg 1$*; then, the errors after the two-cell red-black* μ*-line relaxation will be independent of angle, continuous, and piecewise linear up to accuracy of* $O(\max(\frac{1}{\sigma_t h_{2i-1}}, \frac{1}{\sigma_t h_{2i}}))$.

Proof. See Theorem 3 in [10]. □

In this paper, we will discuss the MLD equation with absorption($\gamma \neq 1$). In the thin limit, γ will not affect the behavior of μ-line relaxation. The proof of Theorem 2 that appears in [10] does not depend on γ. We restate Theorem 2 in [10] for $\gamma \leq 1$ in the following way.

THEOREM 2. *Suppose* $\gamma \leq 1$ *and* $\max(\sigma_t h_{2i-1}, \sigma_t h_{2i}) \ll 1$*; then the errors after two-cell red-black* μ*-line relaxation will be continuous and piecewise linear across two cells up to accuracy of* $O(\max(\sigma_t^2 h_{2i-1}^2, \sigma_t^2 h_{2i}^2))$.

Proof. See the proof of Theorem 2 in [10]. □

In the thick limit, the size of $1-\gamma$ relative to $\frac{1}{\sigma_t h_{2i-1}}$ and $\frac{1}{\sigma_t h_{2i}}$ will determine the effectiveness of μ-line relaxation. For simplicity, we restrict our analysis to uniform $\sigma_t h_i$ and let h be the uniform mesh size. The extension to nonuniform meshes is straightforward but messy. We will analyze the μ-line relaxation when

(a) $1 - \gamma \gg \frac{1}{\sigma_t h}$,

(b) $1 - \gamma = \frac{1}{(\sigma_t h)^p}$, $p = 1, 2, 3$.

In the second case, when $\sigma_t h \gg 1$, the larger p is, the closer γ is to 1.

THEOREM 3. *When* $1 - \gamma \gg \frac{1}{\sigma_t h}$ *and* $\sigma_t h \gg 1$*, the two-cell* μ*-line relaxation alone will reduce errors at any two neighboring cells* $2i - 1, 2i$ *by a factor of* $O(\frac{1}{\sigma_t h})$.

Proof. When $\gamma \neq 1$,

$$\begin{bmatrix} I - \gamma R & -\gamma R \\ -\gamma R & I - \gamma R \end{bmatrix}^{-1} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} + \frac{\gamma}{1-\gamma} \begin{bmatrix} R & R \\ R & R \end{bmatrix}. \tag{3.2}$$

Write

(3.3)

$$\begin{bmatrix} A_{2i-1} & -C_{2i-1} \\ -D_{2i} & A_{2i} \end{bmatrix}$$

$$= \begin{bmatrix} I-\gamma R & -\gamma R & & & & & & \\ -\gamma R & I-\gamma R & & & & & & \\ & & I-\gamma R & -\gamma R & & & & \\ & & -\gamma R & I-\gamma R & & & & \\ & & & & I-\gamma R & -\gamma R & & \\ & & & & -\gamma R & I-\gamma R & & \\ & & & & & & I-\gamma R & -\gamma R \\ & & & & & & -\gamma R & I-\gamma R \end{bmatrix}$$

$$- \begin{bmatrix} B & 0 & -B & 0 & 0 & 0 & 0 & 0 \\ 0 & B & 0 & B & 0 & 0 & 0 & 0 \\ B & 0 & B & 0 & -2B & 0 & 0 & 0 \\ 0 & -B & 0 & B & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & B & 0 & -B & 0 \\ 0 & 0 & 0 & -2B & 0 & B & 0 & B \\ 0 & 0 & 0 & 0 & B & 0 & B & 0 \\ 0 & 0 & 0 & 0 & 0 & -B & 0 & B \end{bmatrix} = H_0 - H_1,$$

where H_0 is the first matrix and H_1 the second in (3.3). By (3.2), H_0 is not singular, so

$$H_0^{-1}H_1 = \frac{1}{\sigma_t h} \begin{bmatrix} M & 0 & -M & 0 & 0 & 0 & 0 & 0 \\ 0 & M & 0 & M & 0 & 0 & 0 & 0 \\ M & 0 & M & 0 & -2M & 0 & 0 & 0 \\ 0 & -M & 0 & M & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & M & 0 & -M & 0 \\ 0 & 0 & 0 & -2M & 0 & M & 0 & M \\ 0 & 0 & 0 & 0 & M & 0 & M & 0 \\ 0 & 0 & 0 & 0 & 0 & -M & 0 & M \end{bmatrix} \tag{3.4}$$

$$+ \frac{\gamma}{(1-\gamma)\sigma_t h} \begin{bmatrix} RM & RM & -RM & RM & 0 & 0 & 0 & 0 \\ RM & RM & -RM & RM & 0 & 0 & 0 & 0 \\ RM & -RM & RM & RM & -2RM & 0 & 0 & 0 \\ RM & -RM & RM & RM & -2RM & 0 & 0 & 0 \\ 0 & 0 & 0 & -2RM & RM & RM & -RM & RM \\ 0 & 0 & 0 & -2RM & RM & RM & -RM & RM \\ 0 & 0 & 0 & 0 & RM & -RM & RM & RM \\ 0 & 0 & 0 & 0 & RM & -RM & RM & RM \end{bmatrix},$$

where $M = \begin{bmatrix} \mu_1 & & \\ & \ddots & \\ & & \mu_n \end{bmatrix}$.

When $\sigma_t h \gg 1$ and $(1-\gamma) \gg \frac{1}{\sigma_t h}$, then $(1-\gamma)\sigma_t h \gg 1$ and we can find a constant c such that

$$||H_0^{-1}H_1||_2 \le \frac{c}{\sigma_t h}. \tag{3.5}$$

So

$$(I - H_0^{-1}H_1)^{-1} = I + H_0^{-1}H_1 + O\left(\frac{1}{\sigma_t^2 h^2}\right). \tag{3.6}$$

Since H_0^{-1} is constant, we have

(3.7)
$$\begin{aligned}\begin{bmatrix} A_{2i-1} & -C_{2i-1} \\ -D_{2i} & A_{2i} \end{bmatrix}^{-1} &= (I - H_0^{-1}H_1)^{-1}H_0^{-1} = H_0^{-1} + H_0^{-1}H_1H_0^{-1} + O\left(\frac{1}{\sigma_t^2 h^2}\right)\\ &= H_0^{-1} + O\left(\frac{1}{\sigma_t h}\right).\end{aligned}$$

From (3.2) and (3.3) we have

$$||H_0^{-1}||_2 \le 1 + \frac{\gamma}{1-\gamma} = \frac{1}{1-\gamma}. \tag{3.8}$$

Therefore, by (3.8) we can find a constant c such that

$$\left\|\begin{bmatrix} A_{2i-1} & -C_{2i-1} \\ -D_{2i} & A_{2i} \end{bmatrix}^{-1}\right\|_2 \le c. \tag{3.9}$$

Substituting (3.9) into (3.1) and noting that $B_i = \frac{1}{\sigma_t h}M$, we have

$$||(e^-_{(2i-1)l}, e^+_{(2i-1)l}, e^-_{(2i-1)r}, e^+_{(2i-1)r}, e^-_{(2i)l}, e^+_{(2i)l}, e^-_{(2i)r}, e^+_{(2i)r})^T||_2 \le c\frac{1}{\sigma_t h}. \tag{3.10}$$

This completes the proof. □

When $\gamma = 1 - \frac{1}{(\sigma_t h)^p}$ and $\sigma_t h \gg 1, p = 1, 2, 3$, the two-cell μ-line relaxation will produce errors that are continuous and essentially independent of angle but kinked across two cells. To show this behavior, we rewrite (3.1) as

$$
\begin{aligned}
\begin{bmatrix} \underline{e}^-_{(2i-1)l} \\ \underline{e}^+_{(2i-1)l} \\ \underline{e}^-_{(2i-1)r} \\ \underline{e}^+_{(2i-1)r} \\ \underline{e}^-_{(2i)l} \\ \underline{e}^+_{(2i)l} \\ \underline{e}^-_{(2i)r} \\ \underline{e}^+_{(2i)r} \end{bmatrix}
&= \begin{bmatrix} A_{2i-1} & -C_{2i-1} \\ -D_{2i} & A_{2i} \end{bmatrix}^{-1}
\begin{bmatrix} 0 \\ 2B_{2i-1} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \underline{e}^+_{(2i-2)r} \\
&\quad + \begin{bmatrix} A_{2i-1} & -C_{2i-1} \\ -D_{2i} & A_{2i} \end{bmatrix}^{-1}
\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 2B_{2i} \\ 0 \end{bmatrix} \underline{e}^-_{(2i+1)l}.
\end{aligned}
\tag{3.11}
$$

When $\sigma_t h \gg 1$ and $\gamma = 1 - \frac{1}{(\sigma_t h)^p}$, we can expand matrix

$$
\begin{bmatrix} A_{2i-1} & -C_{2i-1} \\ -D_{2i} & A_{2i} \end{bmatrix}^{-1}
$$

into Taylor series as $O(\sigma_t h) + O(1) + O(\frac{1}{\sigma_t h})$. We omit the details of the expansion because of the complexity. Let $\delta = \frac{1}{\sigma_t^p h^p}$ and $\epsilon = \frac{1}{\sigma_t h}$. After the expansion, we have

$$
\begin{aligned}
&\begin{bmatrix} A_{2i-1} & -C_{2i-1} \\ -D_{2i} & A_{2i} \end{bmatrix}^{-1}
\begin{bmatrix} 0 \\ 2B_{2i-1} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \\
&= \frac{\xi}{\sigma_t h} \begin{bmatrix} 2RM \\ 2RM \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}
+ \frac{2\xi^2}{\sigma_t^3 h^3} \begin{bmatrix} 0 \\ 0 \\ \frac{4\xi c_0}{\sigma_t h} R \\ \frac{4\xi c_0}{\sigma_t h} R \\ 2R \\ 2R \\ 0 \\ 0 \end{bmatrix}
P(c_0 M^2 R + c_1 M R) + O\left(\frac{1}{\sigma_t h}\right),
\end{aligned}
\tag{3.12}
$$

where

(3.13)
$$\xi = \frac{1}{1-\zeta}, \quad \zeta = \gamma \sum_{j=1}^{n} \left(\frac{2\omega_j (1+\frac{\mu_j}{\sigma_t h})}{1+2\frac{\mu_j}{\sigma_t h}+2(\frac{\mu_j}{\sigma_t h})^2} \right), \quad c_0 = \sum_{j=1}^{n} \omega_j \mu_j, \quad c_1 = \sum_{j=1}^{n} \omega_j \mu_j^2 = \frac{1}{6}$$

and

(3.14)
$$P = \left(1 - \frac{2\epsilon}{\delta + 4c_0\epsilon} MR\right)\left(1 + \frac{2\epsilon}{\delta + 2c_1\epsilon^2} MR\right)\left(1 - \frac{2\epsilon^2(\delta + 2\epsilon c_0)}{8c_0c_1\epsilon^3 + 2c_0\delta\epsilon + \delta^2} M^2 R\right).$$

Here, ξ is not expanded yet. Its expansion will depend on the power p in $\gamma = 1 - \frac{1}{(\sigma_t h)^p}$. We will discuss the three cases $p = 1, 2, 3$ separately.

Case (1) $p = 1$.

When $\sigma_t h \gg 1$, ζ in (3.13) can be expanded as

$$\zeta = \gamma \sum_{j=1}^{n} \left(\frac{2\omega_j (1+\frac{\mu_j}{\sigma_t h})}{1+2\frac{\mu_j}{\sigma_t h}+2(\frac{\mu_j}{\sigma_t h})^2} \right) \tag{3.15}$$

$$= \gamma \sum_{j=1}^{n} 2\omega_j \left(1 + \frac{\mu_j}{\sigma_t h}\right)\left(1 - 2\frac{\mu_j}{\sigma_t h} + O\left(\frac{1}{\sigma_t^2 h^2}\right)\right) = \gamma \left(1 - 2\frac{c_0}{\sigma_t h}\right) + O\left(\frac{1}{\sigma^2 h^2}\right).$$

So by (3.15), ξ in (3.13) can be expanded as

$$\xi = \frac{1}{1-\zeta} = \frac{1}{1-\gamma+\frac{2c_0\gamma}{\sigma_t h}+O(\frac{1}{\sigma_t^2 h^2})} = \frac{\sigma_t h}{1+2c_0} + O\left(\frac{1}{\sigma_t^2 h^2}\right). \tag{3.16}$$

When $p = 1$, P of (3.14) can be expressed as

$$P = (1 + MR)\left(1 - \frac{2}{1+4c_0} MR\right) + O\left(\frac{1}{\sigma_t h}\right). \tag{3.17}$$

By (3.16) and (3.17), the norm of the second term in (3.12) is $O(\frac{1}{\sigma_t h})$. So

$$\begin{bmatrix} A_{2i-1} & -C_{2i-1} \\ -D_{2i} & A_{2i} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 2B_{2i-1} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \frac{1}{1+2c_0} \begin{bmatrix} 2RM \\ 2RM \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + O\left(\frac{1}{\sigma_t h}\right). \tag{3.18}$$

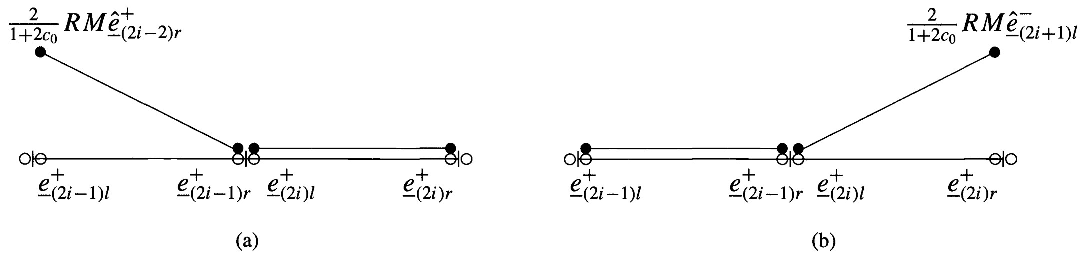

$\frac{2}{1+2c_0}RM\hat{\underline{e}}^{+}_{(2i-2)r}$
$\underline{e}^{+}_{(2i-1)l}$
$\underline{e}^{+}_{(2i-1)r}$
$\underline{e}^{+}_{(2i)l}$
$\underline{e}^{+}_{(2i)r}$
(a)
$\frac{2}{1+2c_0}RM\hat{\underline{e}}^{-}_{(2i+1)l}$
$\underline{e}^{+}_{(2i-1)l}$
$\underline{e}^{+}_{(2i-1)r}$
$\underline{e}^{+}_{(2i)l}$
$\underline{e}^{+}_{(2i)r}$
(b)

FIG. 3.1.

Similarly, we have

$$(3.19)\qquad \begin{bmatrix} A_{2i-1} & -C_{2i-1} \\ -D_{2i} & A_{2i} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 2B_{2i} \\ 0 \end{bmatrix} = \frac{1}{1+2c_0} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 2RM \\ 2RM \end{bmatrix} + O\left(\frac{1}{\sigma_t h}\right).$$

By (3.18) and (3.19), the errors after relaxation will be essentially independent of angle but not piecewise linear. To graphically show this, we plot the errors after the μ-line relaxation in Figure 3.1. Figure 3.1a depicts the first term in (3.11) for positive angles. Figure 3.1b depicts the second term in (3.11) for positive angles. The real errors after relaxation should be a linear combination of Figure 3.1a and b. The error distribution for negative angles will have the same pattern. From Figure 3.1 we see that, after relaxation, $\underline{e}^{+}_{(2i-1)r}$ equals $\underline{e}^{+}_{(2i)l}$ and $\underline{e}^{-}_{(2i-1)r}$ equals $\underline{e}^{-}_{(2i)l}$ up to accuracy of $O(\frac{1}{\sigma_t h})$. We can see that the error after relaxation is kinked and that linear interpolation will not be suitable for this case. However, regardless of the size of $\underline{e}^{+}_{(2i-2)r}$ and $\underline{e}^{+}_{(2i+1)l}$, the errors at the boundary between cell $2i-1$ and cell $2i$ are $O(\frac{1}{\sigma_t h})$.

Case (2) $p = 2$.

When $p = 2$ and $\sigma_t h \gg 1$, with ζ as in (3.15), ξ can be expanded as

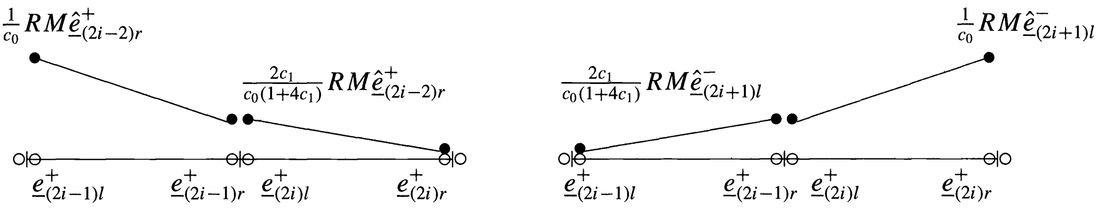
$$(3.20)\qquad \xi = \frac{1}{1-\zeta} = \frac{1}{1-\gamma+\frac{2c_0\gamma}{\sigma_t h}+O(\frac{1}{\sigma_t^2 h^2})} = \frac{\sigma_t h}{2c_0} + O\left(\frac{1}{\sigma_t h}\right)$$

and P in (3.14) can be expanded as

$$(3.21)\qquad P = \frac{\sigma_t h}{1+4c_1}MR + O(1).$$

Substituting (3.20) and (3.21) into (3.12), we have

$$(3.22)\qquad \begin{bmatrix} A_{2i-1} & -C_{2i-1} \\ -D_{2i} & A_{2i} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 2B_{2i-1} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{c_0}RM \\ \frac{1}{c_0}RM \\ \frac{2c_1}{c_0(1+4c_1)}RM \\ \frac{2c_1}{c_0(1+4c_1)}RM \\ \frac{2c_1}{c_0(1+4c_1)}RM \\ \frac{2c_1}{c_0(1+4c_1)}RM \\ 0 \\ 0 \end{bmatrix} + O\left(\frac{1}{\sigma_t h}\right)$$


$\frac{1}{c_0}RM\hat{\underline{e}}^+_{(2i-2)r}$
$\frac{2c_1}{c_0(1+4c_1)}RM\hat{\underline{e}}^+_{(2i-2)r}$
$\underline{e}^+_{(2i-1)l}$
$\underline{e}^+_{(2i-1)r}$
$\underline{e}^+_{(2i)l}$
$\underline{e}^+_{(2i)r}$
$\frac{1}{c_0}RM\hat{\underline{e}}^-_{(2i+1)l}$
$\frac{2c_1}{c_0(1+4c_1)}RM\hat{\underline{e}}^-_{(2i+1)l}$
$\underline{e}^+_{(2i-1)l}$
$\underline{e}^+_{(2i-1)r}$
$\underline{e}^+_{(2i)l}$
$\underline{e}^+_{(2i)r}$

(a) (b)

FIG. 3.2.

and

$$(3.23)\qquad \begin{bmatrix} A_{2i-1} & -C_{2i-1} \\ -D_{2i} & A_{2i} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 2B_{2i+1} \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \frac{2c_1}{c_0(1+4c_1)}RM \\ \frac{2c_1}{c_0(1+4c_1)}RM \\ \frac{2c_1}{c_0(1+4c_1)}RM \\ \frac{2c_1}{c_0(1+4c_1)}RM \\ \frac{1}{c_0}RM \\ \frac{1}{c_0}RM \end{bmatrix} + O\left(\frac{1}{\sigma_t h}\right).$$

We plot (3.22) in Figure 3.2. The errors for negative angles are similar. We can see once again the errors after relaxation are essentially independent of angle and continuous across cells but kinked linear. In fact, they will be sublinear; that is, the error at the interface between cell $2i-1$ and cell $2i$ is less than the average of the errors at the edges of the two-cell pair.

Case (3) $p = 3$.

When $p = 3$ and $\sigma_t h \gg 1$, with ζ in (3.15), ξ can be expanded as

$$(3.24)\qquad \xi = \frac{1}{1-\zeta} = \frac{1}{1-\gamma+\frac{2c_0\gamma}{\sigma_t h}+O(\frac{1}{\sigma_t^2h^2})} = \frac{\sigma_t h}{2c_0} + O\left(\frac{1}{\sigma_t^2 h^2}\right)$$

and P in (3.14) be expanded as

$$(3.25)\qquad P = \frac{\sigma_t h}{4c_1}MR + O(1).$$

Substitute (3.24) and (3.25) into (3.12), we have

$$(3.26)\qquad \begin{bmatrix} A_{2i-1} & -C_{2i-1} \\ -D_{2i} & A_{2i} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 2B_{2i-1} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{c_0}RM \\ \frac{1}{c_0}RM \\ \frac{1}{2c_0}RM \\ \frac{1}{2c_0}RM \\ \frac{1}{2c_0}RM \\ \frac{1}{2c_0}RM \\ 0 \\ 0 \end{bmatrix} + O\left(\frac{1}{\sigma_t h}\right)$$

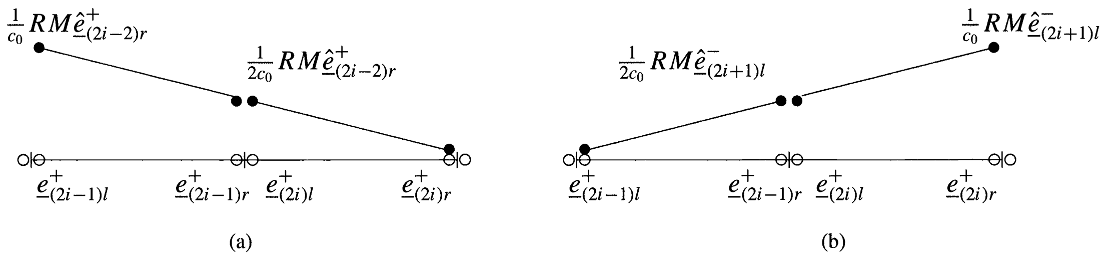

$\frac{1}{c_0}RM\hat{\underline{e}}^+_{(2i-2)r}$
$\frac{1}{2c_0}RM\hat{\underline{e}}^+_{(2i-2)r}$
$\underline{e}^+_{(2i-1)l}$
$\underline{e}^+_{(2i-1)r}$
$\underline{e}^+_{(2i)l}$
$\underline{e}^+_{(2i)r}$
(a)
$\frac{1}{c_0}RM\hat{\underline{e}}^-_{(2i+1)l}$
$\frac{1}{2c_0}RM\hat{\underline{e}}^-_{(2i+1)l}$
$\underline{e}^+_{(2i-1)l}$
$\underline{e}^+_{(2i-1)r}$
$\underline{e}^+_{(2i)l}$
$\underline{e}^+_{(2i)r}$
(b)

FIG. 3.3.

and

$$
\begin{bmatrix} A_{2i-1} & -C_{2i-1} \\ -D_{2i} & A_{2i} \end{bmatrix}^{-1}
\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 2B_{2i+1} \\ 0 \end{bmatrix}
=
\begin{bmatrix} 0 \\ 0 \\ \frac{1}{2c_0}RM \\ \frac{1}{2c_0}RM \\ \frac{1}{2c_0}RM \\ \frac{1}{2c_0}RM \\ \frac{1}{c_0}RM \\ \frac{1}{c_0}RM \end{bmatrix}
+ O\left(\frac{1}{\sigma_t h}\right). \tag{3.27}
$$

We plot (3.26) in Figure 3.3. Again (3.27) is similar. In this case the error after relaxation is independent of angle, linear and continuous across a two-cell pair up to accuracy of $O(\frac{1}{\sigma_t h})$.

In any case, after μ-line relaxation the error across a two-cell pair will be independent of angle, continuous, and sublinear up to order $O(\frac{1}{\sigma_t h})$. The error at the boundary between cell $2i-1$ and cell $2i$ can be expressed as

$$
\underline{e}^{\pm}_{(2i-1)r} = \underline{e}^{\pm}_{(2i)l} + O\left(\frac{1}{\sigma_t h}\right) = \left(\frac{1}{2} - d\right)(\underline{e}^{\pm}_{(2i-1)l} + \underline{e}^{\pm}_{(2i)r}) + O\left(\frac{1}{\sigma_t h}\right), \tag{3.28}
$$

where $0 \leq d \leq \frac{1}{2}$ and ± denotes variables in either positive or negative directions. We summarize the above discussion in the following theorem.

THEOREM 4. *Suppose* $\gamma = 1 - \frac{1}{(\sigma_t h)^p}$ *and* $\sigma_t h \geq 1$. *After two-cell* μ*-line relaxation, the error across a two-cell pair will be independent of angle and continuous up to* $O(\frac{1}{\sigma_t h})$. *Further, the error at the boundary between the two-cell pair can be written as in* (3.28), *where*

(i) $d = \frac{1}{2}$ *for* $p \leq 1$,

(ii) $0 < d < \frac{1}{2}$ *for* $p = 2$,

(iii) $d = 0$ *for* $p \geq 3$.

Proof. The proof follows from the discussion above. □

When $p = 3$, the μ-line relaxation will produce errors of which the dominant terms are piecewise linear. But, in a multigrid algorithm, the mesh size of the coarse grid is twice as large as the fine grid. Since γ is the same on all grids, although $p = 3$ on the finest grid, as the grids go from finest to coarsest level, the relative size of $1 - \gamma$ to $\frac{1}{\sigma_t h}$ will change. To show this, assume that the finest grid is uniform and the number of cells is a power of 2. Now, let $\sigma_t h = 100$ and $\gamma = 1 - \frac{1}{(\sigma_t h)^3} = .999999$; that is, let $p_h = 3$. On the finest level interpolation should be linear. However, four levels down we have $\sigma_t 16h = 1600$ and $\gamma = 0.999999 = 1 - \frac{1}{(\sigma_t 16h)^{1.873}}$; that is, $p_{16h} = 1.873$. From the discussion above we see that linear interpolation is not adequate on this level, even though linear interpolation is sufficient on the finest grid.

We use the parameter d to indicate the severity of deviation of errors from linearity after relaxation. If $\sigma_t h \gg 1$ on the finest level and the relation between γ and $\sigma_t h$ yields $p \leq 1$, then we can set $d = \frac{1}{2}$. If $p \geq 3$ we can set $d = 0$. If $p = 2$ we can use $d = \frac{1}{2(1+4c_1)}$. Unfortunately, the relation between γ and $\sigma_t h$ does not always yield an integer. However, we can use the relaxation to determine the proper interpolation. Consider equation (3.11). Set $\underline{e}^+_{(2i-2)r} = \underline{1}$ and $\underline{e}^+_{(2i+1)l} = \underline{0}$ and use the two-cell inversion to compute the result. Then we may define

$$d_l^{\pm} = \frac{1}{2} \frac{\underline{1}^T(\underline{e}^{\pm}_{(2i-1)l} + \underline{e}^{\pm}_{(2i)r} - (\underline{e}^{\pm}_{(2i-1)r} + \underline{e}^{\pm}_{(2i)l}))}{\underline{1}^T(\underline{e}^{\pm}_{(2i-1)l} + \underline{e}^{\pm}_{(2i)r})}. \tag{3.29}$$

Parameters $d_r^{\pm}$ could be found in a similar manner by setting $\underline{e}^+_{(2i-2)r} = \underline{0}$ and $\underline{e}^+_{(2i+1)l} = \underline{1}$ in (3.11). Of course, for the uniform grid case $d_l^{\pm} = d_r^{\pm}$. Moreover, $d_r^+ = d_r^- + O(\frac{1}{\sigma_t h})$. In practice, we use an average value $d = (d_r^+ + d_r^-)/2$.

Now, suppose we know $\underline{e}^+_{(2i-1)l}$, $\underline{e}^+_{(2i)r}$ and d; then we can approximate $\underline{e}^+_{(2i-1)r}$ and $\underline{e}^+_{(2i)l}$ by interpolation:

$$\begin{bmatrix} \underline{e}^+_{(2i-1)l} \\ \underline{e}^+_{(2i-1)r} \\ \underline{e}^+_{(2i)l} \\ \underline{e}^+_{(2i)r} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \frac{1}{2} - d & \frac{1}{2} - d \\ \frac{1}{2} - d & \frac{1}{2} - d \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \underline{e}^+_{(2i-1)l} \\ \underline{e}^+_{(2i)r} \end{bmatrix}. \tag{3.30}$$

The same interpolation can be used for negative angles. If $\underline{e}^-_{(2i-1)l}$, $\underline{e}^-_{(2i)r}$ and d are known, we have

$$\begin{bmatrix} \underline{e}^-_{(2i-1)l} \\ \underline{e}^-_{(2i-1)r} \\ \underline{e}^-_{(2i)l} \\ \underline{e}^-_{(2i)r} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \frac{1}{2} - d & \frac{1}{2} - d \\ \frac{1}{2} - d & \frac{1}{2} - d \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \underline{e}^-_{(2i-1)l} \\ \underline{e}^-_{(2i)r} \end{bmatrix}. \tag{3.31}$$

For every two-cell pair, the problem is reduced to finding d, $\underline{e}^+_{(2i-1)l}$, $\underline{e}^+_{(2i)r}$, $\underline{e}^-_{(2i-1)l}$, and $\underline{e}^-_{(2i)r}$. If interpolation is relatively accurate, then we can use a coarser grid to approximate $\underline{e}^+_{(2i-1)l}$, $\underline{e}^+_{(2i)r}$, $\underline{e}^-_{(2i-1)l}$, and $\underline{e}^-_{(2i)r}$ on the fine grid. In a multigrid algorithm, a coarse grid solution can be approximated by an even coarser grid, provided a good interpolation can be found. This process can be accomplished recursively down to the coarsest grid. On the coarsest grid, there will be only a few grid points and the solution on the coarsest grid usually can be solved explicitly. For our case, a two-cell μ-line relaxation will exactly solve the coarsest grid, since it has only two cells. Thus, finding a good interpolation is vital to an efficient multigrid scheme.

We define the restriction operator I_h^{2h} by

$$I_h^{2h} = \begin{bmatrix} S_1 & S_2 & & \\ & & \ddots & \\ & & & S_1 \quad S_2 \end{bmatrix} \otimes I, \tag{3.32}$$

where I is the $N \times N$ identity and

$$S_1 = \begin{bmatrix} \frac{1}{2} & 0 & \frac{1}{4} & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{4} \\ 0 & 0 & \frac{1}{4} & 0 \\ 0 & 0 & 0 & \frac{1}{4} \end{bmatrix}, \quad S_2 = \begin{bmatrix} \frac{1}{4} & 0 & 0 & 0 \\ 0 & \frac{1}{4} & 0 & 0 \\ \frac{1}{4} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{4} & 0 & \frac{1}{2} \end{bmatrix}. \tag{3.33}$$

Note that the restriction operator represents full weighting. We also define the interpolation I_{2h}^h as

$$I_{2h}^h = \begin{bmatrix} T_1 & & \\ T_2 & & \\ & \ddots & \\ & & T_1 \\ & & T_2 \end{bmatrix} \otimes I, \tag{3.34}$$

where I is the $N \times N$ identity and
(3.35)

$$T_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \frac{1}{2}-d & 0 & \frac{1}{2}-d & 0 \\ 0 & \frac{1}{2}-d & 0 & \frac{1}{2}-d \end{bmatrix}, \quad T_2 = \begin{bmatrix} \frac{1}{2}-d & 0 & \frac{1}{2}-d & 0 \\ 0 & \frac{1}{2}-d & 0 & \frac{1}{2}-d \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

When $d = 0$, the interpolation operator I_{2h}^h is the transpose of the restriction operator scaled by a factor of $\frac{1}{2}$.

Generalization to a nonuniform grid is straightforward. Suppose a two-cell pair has cell widths $h_{(2i-1)}$ and $h_{(2i)}$. A two-cell inversion in (3.11) with $\underline{e}^+_{(2i-2)r} = \underline{1}$ and $\underline{e}^+_{(2i+1)l} = \underline{0}$ yields the deviation from linearity, which is then built into the interpolation formulas. One could carry a different interpolation for r and l as well as $+$ and $-$ if not in the asymptotic regime. With a uniform grid, the computation need only be done once per grid level, while for the nonuniform grid it must be done for each two-cell pair. However, all two-cell pairs can be calculated in parallel. The restriction operators correspond to full weighting, that is, the transpose of linear interpolation is weighted so that each row sums to 1.

We remark that the two-cell μ-line relaxation can be adapted to the context of mildly anisotropic scattering. For example, if the scattering operator involves P_3 scattering, then it has only four nonzero eigenvalues. This will result in a two-cell problem involving an easily solved linear system plus a rank 16 matrix. The two-cell inversion will require the solution of two systems involving a 16×16 matrix. However, the matrix will be the same throughout a particular material and could be factored before the start of the computation. The total computation would be $O(N)$.

In the next section, we will discuss the multigrid algorithm using the kinked interpolation operator I_{2h}^h and restriction operator I_h^{2h}.

4. Multigrid algorithm.

In this section, we derive the coarse grid operator and show that, although the coarse grid operator no longer represents MLD on the coarse grid, it has the same structure as the fine grid operator, and two-cell μ-line relaxation can be accomplished with the same formula and cost.

From (2.4), we define the fine grid operator L^h as

$$L^h = \begin{bmatrix} A^h & -C^h & & \\ -D^h & A^h & \ddots & \\ & \ddots & \ddots & -C^h \\ & & -D^h & A^h \end{bmatrix}, \tag{4.1}$$

where A^h, C^h, and B^h are defined by (2.5) for uniform grid h. With restriction operator I_h^{2h} and interpolation operator I_{2h}^h defined by (3.32) and (3.35), the coarse grid operator L^{2h} can be obtained by

$$L^{2h} = I_{2h}^h L^h I_h^{2h} = \begin{bmatrix} A^{2h} & -C^{2h} & & \\ -D^{2h} & A^{2h} & \ddots & \\ & \ddots & \ddots & -C^{2h} \\ & & -D^{2h} & A^{2h} \end{bmatrix}, \tag{4.2}$$

where

$$A^{2h} = \begin{bmatrix} S_1 & S_2 \end{bmatrix} \begin{bmatrix} A^h & -C^h \\ -D^h & A^h \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix}, \tag{4.3a}$$

$$C^{2h} = \begin{bmatrix} S_1 & S_2 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ C^h & 0 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix}, \tag{4.3b}$$

$$D^{2h} = \begin{bmatrix} S_1 & S_2 \end{bmatrix} \begin{bmatrix} 0 & D^h \\ 0 & 0 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix}. \tag{4.3c}$$

It is easy to verify that (see (2.1f), (2.5b), and (3.4))

$$C^{2h} = \frac{1}{\sigma_t h} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ M & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{4.4a}$$

$$D^{2h} = \frac{1}{\sigma_t h} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & M \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{4.4b}$$

To derive A^{2h}, we write A^h as

$$A^h = \begin{bmatrix} F^h & 0 & G^h & 0 \\ 0 & K^h & 0 & Z^h \\ Z^h & 0 & K^h & 0 \\ 0 & G^h & 0 & F^h \end{bmatrix} - \begin{bmatrix} a^hR & a^hR & b^hR & b^hR \\ a^hR & a^hR & b^hR & b^hR \\ b^hR & b^hR & a^hR & a^hR \\ b^hR & b^hR & a^hR & a^hR \end{bmatrix}, \tag{4.5}$$

where F^h, G^h, K^h, and Z^h are diagonal matrices and a^h and b^h are scalars. We use this notation to explain that the coarse grid operator will follow the same pattern as the fine grid operator; that is, A^h and consecutive two-cell coarse grid operators A^{2h}, A^{4h} ... are all composed of a matrix whose components are diagonal matrices plus a rank two matrix. For the fine grid h, $F^h = I + B$, $G^h = -B$, $K^h = I + B$, $Z^h = B$, $a^h = 1$, and $b^h = 0$. After multiplication, (4.3a) can be written as

$$A^{2h} = \begin{bmatrix} F^{2h} & 0 & G^{2h} & 0 \\ 0 & K^{2h} & 0 & Z^{2h} \\ Z^{2h} & 0 & K^{2h} & 0 \\ 0 & G^{2h} & 0 & F^{2h} \end{bmatrix} - \begin{bmatrix} a^{2h}R & a^{2h}R & b^{2h}R & b^{2h}R \\ a^{2h}R & a^{2h}R & b^{2h}R & b^{2h}R \\ b^{2h}R & b^{2h}R & a^{2h}R & a^{2h}R \\ b^{2h}R & b^{2h}R & a^{2h}R & a^{2h}R \end{bmatrix}, \tag{4.6}$$

where

$$F^{2h} = \frac{1}{4}[(2.5 - d)F^h + (1 - 2d)G^h + (0.5 - d)K^h + Z^h - (1 - 2d)B], \tag{4.7a}$$

$$G^{2h} = \frac{1}{4}[(0.5 - d)F^h + (2 - 2d)G^h + (0.5 - d)K^h - (1 - 2d)B], \tag{4.7b}$$

$$K^{2h} = \frac{1}{4}[(0.5 - d)F^h + G^h + (2.5 - d)K^h + (1 - 2d)Z^h - (1 - 2d)B], \tag{4.7c}$$

$$Z^{2h} = \frac{1}{4}[(0.5 - d)F^h + (0.5 - d)K^h + (2 - 2d)Z^h - (1 - 2d)B], \tag{4.7d}$$

$$a^{2h} = 0.75a^h - \frac{d}{2}a^h + 0.5b^h - \frac{d}{2}b^h, \tag{4.7e}$$

$$b^{2h} = 0.25a^h - \frac{d}{2}a^h + 0.5b^h - \frac{d}{2}b^h. \tag{4.7f}$$

Since F^h, G^h, K^h, Z^h, and B are diagonal matrices, then F^{2h}, G^{2h}, K^{2h}, Z^{2h} are also diagonal matrices. Structurally, A^h and A^{2h} are the same. For two-cell μ-line relaxation on grid $2h$, we can still easily invert

$$\begin{bmatrix} A^{2h}_{2i-1} & -C^{2h}_{2i-1} \\ -D^{2h}_{2i} & A^{2h}_{2i} \end{bmatrix}$$

with $O(n)$ operations by taking advantage of the fact that the matrix to be inverted consists of an easily invertible matrix with diagonal components and a rank four matrix. The coarse grid operator on grid $4h$ also will have the same structure as on grids h and $2h$.

The generalization to nonuniform grids is again straightforward. Each two-cell pair will yield a different A^{2h}, but it will again have the form as in (4.6) and two-cell inversion on each coarse grid will be amenable to fast inversion.

In the next section, we will display properties of the kinked interpolation operator.

5. Properties of kinked interpolation. In this section, we will display the properties of the kinked interpolation operator by solving (2.5) with boundary conditions

$$\underline{\psi}^+_{0r} = \underline{1}, \quad \underline{\psi}^-_{(m+1)l} = \underline{0} \tag{5.1}$$

on a slab of $x\epsilon(-1, 1)$. Suppose h is the mesh size of the finest grid and H is the mesh size of the coarsest grid, which has only two cells. We choose $\sigma_t \gg 1$ and

$$\gamma = 1 - \frac{1}{\sigma_t^q} \tag{5.2}$$

TABLE 5.1
Value of p on various grid levels.

m	128	64	32	16	8	4	2
$q = 1$	10.32	4.04	2.51	1.82	1.43	1.17	1
$q = 2$	20.64	8.08	5.02	3.64	2.86	2.34	2

for $q = 1, 2, 3$. The case $q = 1$ represents substantial absorption. The case $q = 2$ is the case of most interest to the transport community and yields the well-known thick diffusion limit. The case $q = 3$ also yields the thick diffusion limit but with little absorption.

Given σ_t and γ and grid size h, the value p that satisfies

$$\gamma = 1 - \frac{1}{(\sigma_t h)^p} \tag{5.3}$$

describes the behavior of the error after relaxation as in Theorem 4. This analysis assumes $\sigma_t h \gg 1$. Equating (5.2) and (5.3) yields

$$p = q\frac{\log(\sigma_t)}{\log(\sigma_t h)}. \tag{5.4}$$

In our analysis, $h < 1$, which implies $p \geq q$ on all grids with $p = q$ on the coarsest grid where $H = 1$. This shows that if $q = 3$, the error after relaxation will be linear on all grids. It is only for $q < 3$ that kinked elements are necessary. To see this, let $\sigma_t = 100$. Table 5.1 shows the values of p for $q = 1$ and $q = 2$ on grids with m varying from 128 to 2.

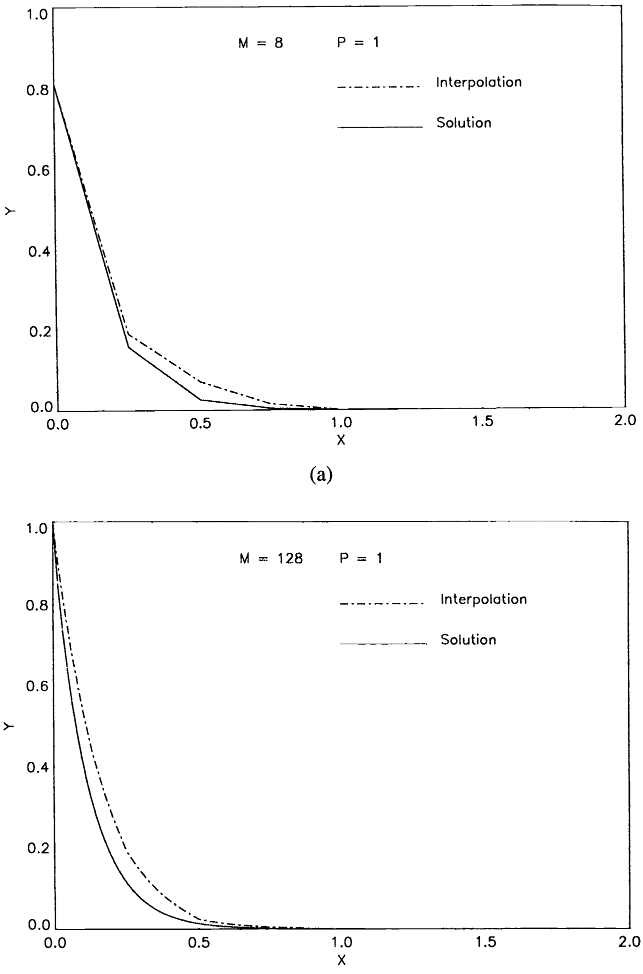
For each set of σ_t, γ, we can find parameter d defined in the previous section on each level (from finest grid to coarsest grid). With d on every grid obtained, we have kinked interpolation on each grid. We define the accumulated interpolation as

$$I_H^h = I_{2h}^h I_{4h}^{2h} \cdots\cdots\cdots I_H^{H/2}. \tag{5.5}$$

With accumulated interpolation so defined, we then solve a homogeneous S_8 transport equation with unit isotropic boundary condition on the left and zero on the right on grids with 8 cells and 128 cells, respectively. We choose $\sigma_t = 100$ and $\gamma = 1 - \frac{1}{\sigma_t^p}$, $p = 1, 2, 3$. We choose an angle line $\mu = 0.525$ for the comparison of the interpolated approximation and the exact discretized solution. With the calculated values $\underline{\psi}_{1l}^+$ and $\underline{\psi}_{mr}^+$, we use the accumulated kinked interpolation operator I_H^h from the coarsest grid to the finest grid to obtain all interior values on the finest grid and compare the interpolated approximation with the calculated solution. Figure 5.1 depicts the case in which there is substantial absorption ($p = 1$). In this case, the relaxation itself is very efficient.

Figure 5.2 shows the case that is of most interest in the transport community ($p = 2$). Figure 5.3 shows the case in which there is not much absorption ($p = 3$). Notice how well interpolation approximates the exact MLD on the finest grid. From Figure 5.3 we can see that when the power p increases, the interpolation is more accurate since the kinked elements are less kinked. We can see from these plots that the relaxation-induced interpolation we have designed is accurate and suitable for the multigrid algorithm.

In the next section, we will present computational results of the multigrid algorithm using kinked interpolation.



(a)

(b)

FIG. 5.1.

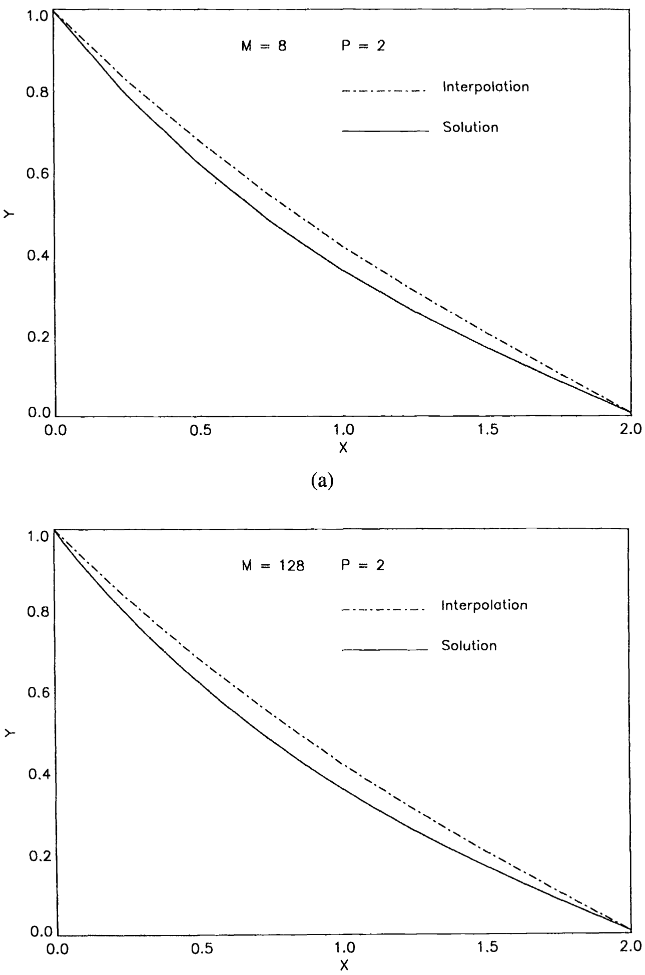
6. Computational results.

6.1. Multigrid convergence factors. The following computational results were conducted on the domain $x \in [0,\ 1]$ using homogeneous boundary conditions. The domain was divided into $m = 128$ spatial cells, and an S_{32} discretization ($n = 16$) was used in the angle. The problem is then specified completely by choosing $\sigma_t h$ and $\gamma = \frac{\sigma_s}{\sigma_t}$. Convergence factors were computed by setting the right-hand side equal to zero, choosing a random initial guess, and performing 15 $V(1, 1)$ cycles and taking the geometric mean of the last 5 cycles. This process exposes the most slowly converging eigen components. Initial reduction is usually much faster.

From computational results conducted on S_N with N ranging from 2 to 256, we observe the same convergence performance as those shown in Table 6.1.

From Table 6.1, we observe that the convergence factor ρ is $O(\sigma_t^3 h^3)$ when $\sigma_t h \ll 1$.

For the thick limit, we will let $\gamma = 1 - (\frac{1}{\sigma_t h})^p$, since, from our previous section, the power p will influence the error distribution after relaxation. We will present results with $p = 1, 2, 3$, and ∞.



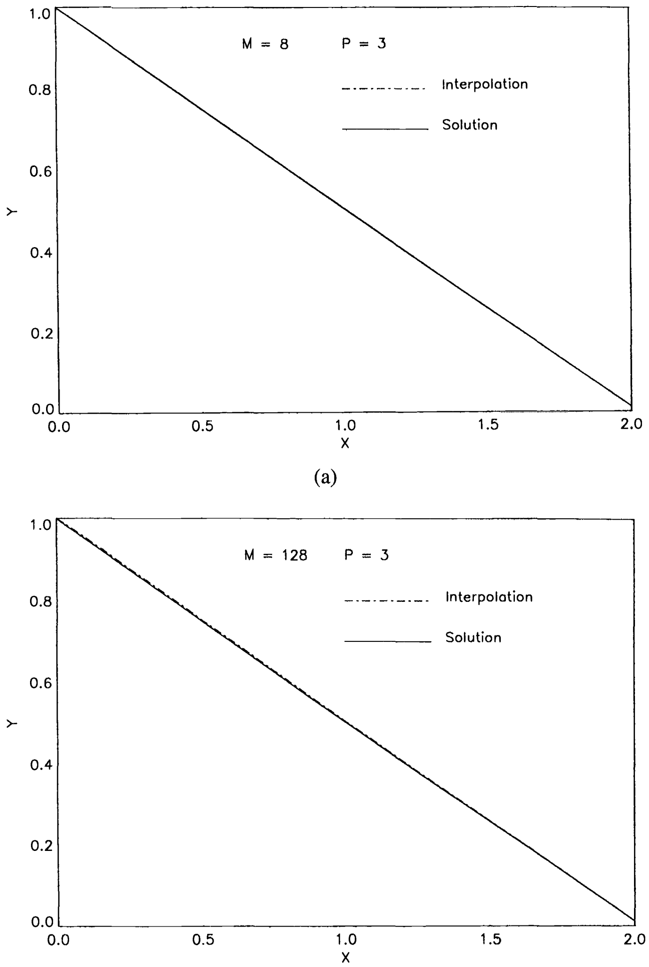
(a)

(b)

FIG. 5.2.

In Table 6.2, the convergence rates ρ are for a multigrid $V(1,1)$ cycle with linear interpolation. Table 6.3 contains the convergence rates for a multigrid $V(1,1)$ cycle with kinked interpolation.

Table 6.2 shows that the multigrid $V(1, 1)$ cycle with linear interpolation has a convergence factor ρ on the order of $O((\frac{1}{\sigma_t h})^2)$ with $\gamma = 1$ ($p = \infty$) in the thick limit. But, when there is absorption ($\gamma \neq 1$), the convergence factor for linear interpolation is not satisfactory at all. As we have discussed in previous sections, when $\gamma \neq 1$ the errors after relaxation will be kinked. Linear interpolation will not be suitable for this case since the linear interpolation will simply interpolate the coarse grid approximation to the fine grid as if the error on the fine grid after relaxation were linear. So, even if the coarse grid is solved exactly, linear interpolation of this solution to the fine grid will not approximate the error on the fine grid. The situation will be compounded for a multigrid $V(1, 1)$ cycle since there are many levels. Every coarse grid solution will not approximate the error on the next finer grid. From our discussion in the previous section, when $p = 3$ on the fine grid, there is relatively little absorption and the errors after relaxation will be nearly linear. But, as we go down to coarser grids, we will reach a level



(a)

(b)

FIG. 5.3.

TABLE 6.1
Convergence factors for uniform grid.

$\sigma_t h$	$\gamma = 0.999999$	$\gamma = 0.999$	$\gamma = 0.9$	$\gamma = 0.7$
10^{-5}	0.37×10^{-9}	0.37×10^{-9}	0.35×10^{-9}	0.33×10^{-9}
10^{-4}	0.31×10^{-6}	0.31×10^{-6}	0.31×10^{-6}	0.30×10^{-6}
10^{-3}	0.11×10^{-3}	0.11×10^{-3}	0.11×10^{-3}	0.9×10^{-4}
10^{-2}	0.50×10^{-2}	0.50×10^{-2}	0.44×10^{-2}	0.38×10^{-2}
10^{-1}	0.68×10^{-2}	0.43×10^{-2}	0.68×10^{-2}	0.68×10^{-2}
10^{-0}	0.70×10^{-2}	0.72×10^{-2}	0.14×10^{-2}	0.19×10^{-2}

on which $\sigma_t h$ will satisfy $\gamma = 1 - (\frac{1}{\sigma_t h})^2$. Thus, using linear interpolation on this particular level will adversely affect the overall convergence factor. From Table 6.3 we see that when kinked interpolation is used the multigrid $V(1,1)$ has a convergence factor ρ on the order of $O(\frac{1}{\sigma_t h})$ for all levels of absorption. When there is no absorption, the multigrid $V(1,1)$ with kinked interpolation has a convergence factor on the order of $O((\frac{1}{\sigma_t h})^2)$.

TABLE 6.2
Convergence factors for linear interpolation.

$\sigma_t h$	$p = 1$	$p = 2$	$p = 3$	$p = \infty$
10^1	0.37×10^{-2}	0.15	0.32	0.27×10^{-4}
10^2	0.78×10^{-2}	0.61	0.74	0.22×10^{-6}
10^3	0.84×10^{-2}	0.82	0.62	0.20×10^{-8}
10^4	0.85×10^{-2}	0.85	0.34	0.18×10^{-10}

TABLE 6.3
Convergence factors for kinked interpolation.

$\sigma_t h$	$p = 1$	$p = 2$	$p = 3$	$p = \infty$
10^1	0.22×10^{-5}	0.43×10^{-2}	0.53×10^{-2}	0.85×10^{-4}
10^2	0.45×10^{-6}	0.25×10^{-2}	0.51×10^{-3}	0.11×10^{-5}
10^3	0.47×10^{-7}	0.36×10^{-3}	0.89×10^{-5}	0.13×10^{-7}
10^4	0.49×10^{-8}	0.39×10^{-4}	0.11×10^{-6}	0.14×10^{-9}

TABLE 6.4
Convergence factors for kinked interpolation, $\gamma = 0.999$.

σ_t	240	245	250	256	260	265	270	275	280
ρ	0.0082	0.0083	0.0084	0.0085	0.0085	0.0085	0.0084	0.0084	0.0083

TABLE 6.5
Convergence factors for nonuniform grid.

c	$\gamma = 0.999999$	$\gamma = 0.999$
10^{-4}	0.12×10^{-10}	0.12×10^{-10}
10^{-3}	0.32×10^{-7}	0.32×10^{-7}
10^{-2}	0.29×10^{-4}	0.29×10^{-4}
10^{-1}	0.20×10^{-4}	0.10×10^{-2}
10^{-0}	0.22×10^{-3}	0.32×10^{-2}
10^1	0.17×10^{-1}	0.54×10^{-2}
10^2	0.70×10^{-1}	0.40×10^{-3}
10^3	0.78×10^{-1}	0.19×10^{-3}
10^4	0.45×10^{-1}	0.17×10^{-4}
10^5	0.19×10^{-2}	0.20×10^{-5}

In Table 6.4, we show a range within which the worst convergence factor occurs when $\gamma = 0.999$ and $m = 256$. In this case, the maximal ρ is 0.0085.

In Table 6.5, we present convergence factors of the multigrid $V(1,1)$ cycle with kinked interpolation for a nonuniform grid. We select $\sigma_t h_i = c \times 10^{2*\eta_i}$, where η_i is a random number between $(-1,1)$. For example, when $c = 1$, $\sigma_t h$ ranges from 0.01 to 100. We use parameter c to shift from the thin limit to the thick limit.

From Table 6.5, we see that the kinked interpolation is also suitable for nonuniform grid or, equivalently, for nonconstant σ_t.

6.2. DSA convergence factors. In this section, we will present a comparison of the DSA algorithm [1, 2, 5] and the multigrid algorithm with kinked interpolation. The slab is assumed to have physical thickness 2. Thus, σ_t represents the width of the slab measured in the number of mean-free paths. The tests were performed using S_8 and a wide range of σ_t and m (the

TABLE 6.6
Convergence factors for multigrid $\gamma = 0.999$.

σ_t	$m = 16$	$m = 64$	$m = 256$	$m = 1024$
4^{-5}	0.0	0.0	0.0	0.0
4^{-4}	0.32×10^{-10}	0.13×10^{-10}	0.10×10^{-10}	0.10×10^{-10}
4^{-3}	0.71×10^{-8}	0.30×10^{-8}	0.19×10^{-8}	0.20×10^{-8}
4^{-2}	0.12×10^{-5}	0.51×10^{-6}	0.30×10^{-6}	0.30×10^{-6}
4^{-1}	0.61×10^{-4}	0.29×10^{-4}	0.13×10^{-4}	0.11×10^{-4}
1	0.57×10^{-3}	0.37×10^{3}	0.32×10^{-3}	0.31×10^{-3}
4	0.12×10^{-2}	0.12×10^{-2}	0.11×10^{-2}	0.13×10^{-1}
4^2	0.31×10^{-2}	0.36×10^{-2}	0.54×10^{-2}	0.76×10^{-2}
4^3	0.57×10^{-2}	0.79×10^{-2}	0.73×10^{-2}	0.11×10^{-1}
4^4	0.57×10^{-2}	0.69×10^{-2}	0.85×10^{-2}	0.78×10^{-2}
4^5	0.68×10^{-3}	0.58×10^{-3}	0.71×10^{-2}	0.86×10^{-2}
4^6	0.93×10^{-5}	0.71×10^{-3}	0.58×10^{-2}	0.71×10^{-2}
4^7	0.25×10^{-7}	0.95×10^{-5}	0.72×10^{-3}	0.58×10^{-2}
4^8	0.33×10^{-10}	0.25×10^{-7}	0.94×10^{-5}	0.72×10^{-3}
4^9	0.0	0.32×10^{-10}	0.25×10^{-7}	0.94×10^{-5}
4^{10}	0.0	0.0	0.32×10^{-10}	0.25×10^{-7}

TABLE 6.7
Convergence factors for DSA $\gamma = 0.999$.

σ_t	$m = 16$	$m = 64$	$m = 256$	$m = 1024$
4^{-5}	0.19×10^{-2}	0.19×10^{-2}	0.190×10^{-2}	0.190×10^{-2}
4^{-4}	0.75×10^{-2}	0.75×10^{-2}	0.75×10^{-2}	0.75×10^{-2}
4^{-3}	0.28×10^{-1}	0.28×10^{-1}	0.28×10^{-1}	0.28×10^{-1}
4^{-2}	0.83×10^{-1}	0.83×10^{-1}	0.83×10^{-1}	0.84×10^{-1}
4^{-1}	0.153	0.150	0.150	0.151
1	0.186	0.203	0.204	0.204
4	0.179	0.216	0.216	0.216
4^2	0.142	0.165	0.204	0.210
4^3	0.127	0.142	0.149	0.163
4^4	0.118	0.129	0.143	0.156
4^5	0.102	0.119	0.130	0.144
4^6	0.66×10^{-1}	0.102	0.119	0.130
4^7	0.27×10^{-1}	0.66×10^{-1}	0.102	0.119
4^8	0.82×10^{-2}	0.27×10^{-1}	0.66×10^{-1}	0.102
4^9	0.21×10^{-2}	0.82×10^{-2}	0.27×10^{-1}	0.66×10^{-1}
4^{10}	0.54×10^{-3}	0.21×10^{-2}	0.82×10^{-2}	0.27×10^{-1}

number of cells) and various values of γ ranging from .9999 to .9. The performance was similar for each value of γ, so we present only the complete results for $\gamma = .999$. The results for other values of γ can be found in [11]. Tables 6.6 and 6.7 give convergence factors for multigrid and DSA for various values of σ_t and m. The diagonals of these tables represent constant $\sigma_t h$.

In Tables 6.6 through 6.9, we say the convergence factor is zero if $\rho < 10^{-11}$. For both algorithms, the convergence factors are roughly constant along diagonals. But for each fixed γ, the multigrid convergence factor will approach zero much faster than the DSA convergence factor as σ_t goes to ∞ in the thick limit and σ_t goes to zero in the thin limit.

TABLE 6.8
Adjusted convergence factors for DSA $\gamma = 0.999$.

σ_t	$m = 16$	$m = 64$	$m = 256$	$m = 1024$
4^{-5}	0.16×10^{-6}	0.16×10^{-6}	0.16×10^{-6}	0.16×10^{-6}
4^{-4}	0.49×10^{-5}	0.49×10^{-5}	0.49×10^{-5}	0.49×10^{-5}
4^{-3}	0.13×10^{-3}	0.13×10^{-3}	0.13×10^{-3}	0.13×10^{-3}
4^{-2}	0.20×10^{-2}	0.20×10^{-2}	0.20×10^{-2}	0.20×10^{-2}
4^{-1}	0.92×10^{-2}	0.87×10^{-2}	0.87×10^{-2}	0.87×10^{-2}
1	0.15×10^{-1}	0.19×10^{-1}	0.19×10^{-1}	0.19×10^{-1}
4	0.14×10^{-1}	0.22×10^{-1}	0.22×10^{-1}	0.22×10^{-1}
4^2	0.76×10^{-2}	0.11×10^{-1}	0.19×10^{-1}	0.22×10^{-1}
4^3	0.57×10^{-2}	0.76×10^{-2}	0.86×10^{-2}	0.11×10^{-1}
4^4	0.48×10^{-2}	0.59×10^{-2}	0.77×10^{-2}	0.96×10^{-2}
4^5	0.33×10^{-2}	0.49×10^{-2}	0.61×10^{-2}	0.79×10^{-2}
4^6	0.12×10^{-2}	0.33×10^{-2}	0.49×10^{-2}	0.61×10^{-2}
4^7	0.12×10^{-3}	0.11×10^{-2}	0.33×10^{-2}	0.49×10^{-2}
4^8	0.61×10^{-5}	0.12×10^{-3}	0.11×10^{-2}	0.33×10^{-2}
4^9	0.20×10^{-6}	0.61×10^{-5}	0.12×10^{-3}	0.11×10^{-2}
4^{10}	0.68×10^{-8}	0.20×10^{-6}	0.61×10^{-5}	0.12×10^{-3}

In all cases, the multigrid convergence factors were superior to the DSA convergence factors. However, it is important to adjust for the relative amount of computational work required by each algorithm. Of course, such measures will be machine dependent. On the Cray Y/MP, where these tests were performed, we compared times for $N = 64$ and $m = 1024$. Ten $V(1,1)$ cycles required 93 seconds. This includes 4.2 seconds to obtain the parameter d on every level. The parameter d on the coarse grid will depend on the relaxation on the fine grid. Thus, this setup process is sequential. One $V(1,1)$ cycle required 9.3 seconds and one DSA cycle required 3.7 seconds. The ratio of these times is 2.5. Table 6.8 contains the results of raising each entry in Table 6.7 to the power 2.5. This is a fairer comparison with the multigrid tables on a serial machine.

From Tables 6.6–6.8 we see that the convergence factors are nearly constant along diagonals, that is, for constant $\sigma_t h$. This was true for other values of γ also. In Table 6.9 we present both multigrid and adjusted DSA convergence factors for $m = 1024$, four values of γ, and various values of $\sigma_t h$. Again, we see that multigrid is faster in all tests. The two algorithms have nearly equal rates in regimes in which $\sigma_t h \sim 1$. This is the region in which the relaxation will not produce continuous kinked linear errors across two cells and, thus, the kinked interpolation is not as accurate.

On a parallel machine we expect the results to more heavily favor the multigrid algorithm. Both algorithms can be implemented with parallel complexity $O(\log(m))$. In this context, however, we expect the times to be more nearly equal. The multigrid algorithm has been implemented on an Thinking Machines Inc. CM-200 [8, 9]. The fundamental step in the DSA algorithm, the transport sweep, has also been implemented on the CM-200. A form of cyclic reduction was used. A comparison of timings on this machine appears in [15].

In either setting, parallel or serial, a full multigrid algorithm (FMG) can be implemented [3, 12]. This would provide a savings in some regions of the tables. Moreover, full multigrid provides a natural framework for adaptive grid refinement.

7. Conclusions. From our analysis and computational results, we conclude the following:

TABLE 6.9
Convergence factors for multigrid/adjusted DSA.

$\sigma_t h$	$\gamma = .9999$	$\gamma = .999$	$\gamma = .99$	$\gamma = .9$
2^{-9}	3.2e−04 / 1.9e−02	3.1e−04 / 1.9e−02	3.5e−04 / 1.9e−02	3.2e−04 / 1.5e−02
2^{-7}	1.1e−03 / 2.2e−02	1.3e−02 / 2.2e−02	3.2e−03 / 2.2e−02	1.5e−03 / 1.7e−02
2^{-5}	1.3e−03 / 2.1e−02	7.6e−03 / 2.2e−02	9.8e−03 / 2.1e−02	2.0e−03 / 1.5e−02
2^{-3}	7.1e−03 / 1.2e−02	1.1e−02 / 1.1e−02	1.0e−02 / 4.1e−02	2.3e−03 / 1.2e−02
2^{-1}	3.9e−03 / 8.3e−02	7.8e−03 / 9.6e−03	1.0e−02 / 8.7e−02	7.6e−04 / 5.1e−03
2	2.6e−03 / 7.7e−03	8.6e−03 / 7.9e−03	4.5e−03 / 6.9e−03	3.5e−04 / 2.9e−03
2^3	4.1e−03 / 6.2e−03	7.1e−03 / 6.1e−03	8.4e−04 / 4.4e−03	9.0e−05 / 5.6e−04
2^5	4.0e−03 / 5.5e−03	5.8e−03 / 4.9e−03	4.5e−04 / 1.8e−03	2.2e−07 / 3.8e−05
2^7	2.3e−03 / 5.2e−03	7.2e−04 / 3.3e−03	2.0e−06 / 2.8e−04	2.7e−10 / 1.6e−06
2^9	4.8e−04 / 4.4e−03	9.4e−06 / 1.1e−03	3.1e−09 / 1.6e−05	0 / 5.0e−08
2^{11}	2.1e−05 / 2.6e−03	2.5e−08 / 1.2e−04	3.0e−12 / 6.7e−08	0 / 1.7e−09
2^{13}	1.5e−07 / 5.9e−04	3.2e−11 / 6.1e−06	0 / 2.2e−08	0 / 5.5e−11
2^{15}	2.7e−10 / 4.3e−05	0 / 2.0e−07	0 / 7.2e−10	0 / 0
2^{17}	0 / 1.9e−06	0 / 6.8e−09	0 / 2.1e−11	0 / 0

- The two-cell μ-line red-black relaxation is inexpensive, highly vectorizable, and parallelizable. Modification of this algorithm to incorporate mildly anisotropic scattering is also viable.
- When there is no absorption ($\gamma = 1$), this relaxation will cause the error to be linear, independent of angle, and continuous across two cells up to $O(\frac{1}{\sigma_t h})$ accuracy when $\sigma_t h \gg 1$.
- When $\gamma < 1$, the error after relaxation will be essentially kinked linear. The severity of deviation from linearity of the error after relaxation is determined by the values γ, $\sigma_t h$, and p in $\gamma = 1 - (\frac{1}{\sigma_t h})^p$. In the thick limit, the error after relaxation will be independent of angle up to accuracy of $O(\frac{1}{\sigma_t h})$.
- Relaxation-induced kinked interpolation is fairly accurate, and it can be obtained by the relaxation operator on each level to ensure that each level will have a good coarse grid approximation. This methodology can be extended to higher dimensions.
- The error reduction factor ρ for a multigrid $V(1,1)$ cycle is $\rho = O((\sigma_t h)^3)$ for the thin limit and $\rho = O(\frac{1}{\sigma_t h})$ for the thick limit when $\gamma = 1 - (\frac{1}{\sigma_t h})^p$ and $1 < p < 3$. When p is smaller than 1, there will be more absorption and the multigrid performance will be better. When p is more than 3, there will be hardly any absorption and the multigrid performance will be close to that of the performance when $\gamma = 1$ [10].
- The multigrid scheme was shown computationally to be effective for highly irregular meshes and heterogeneous material.
- The multigrid algorithm is more efficient than the DSA algorithm in all regimes.

REFERENCES

[1] M. L. ADAMS and W. R. MARTIN, *Diffusion-synthetic acceleration of discontinuous finite-element transport iterations*, Nuclear Sci. Eng, to appear.

[2] R. E. ALCOUFFE, *Diffusion synthetic acceleration methods for the diamond-differenced discrete-ordinates equations*, Nuclear Sci. Eng., 64 (1997), p. 344.

[3] A. BRANDT, *Multi-level adaptive solution to boundary-value problems*, Math. Comp., 31 (1977), pp. 333–390.

[4] V. FABER AND T. A. MANTEUFFEL, *Neutron transport from the viewpoint of linear algebra*, in Transport Theory, Invariant Imbedding and Integral Equations, Lecture Notes in Pure and Applied Mathematics, 115, P. Nelson et al., eds., Marcel-Dekker, New York, 1989, pp. 37–61.

[5] E. W. Larsen, *Unconditionally stable diffusion-synthetic acceleration methods for the slab geometry discrete ordinates equations*, Nuclear Sci. Eng., 82 (1982), p. 47.

[6] E. W. Larsen and J. E. Morel, *Asymptotic solutions of numerical transport problems in optically thick diffusive regimes* II, J. Comp. Phys., 83 (1989), p. 212.

[7] E. E. Lewis and W. F. Miller, Jr., *Computational Methods of Neutron Transport*, 1984.

[8] T. A. Manteuffel, S. McCormick, J. Morel, S. Oliveira, and G. Yang, *Parallel multigrid methods for transport equations*, in Proc. Copper Mountain Conference on Iterative Methods, Copper Mountain, CO, April 9–14, 1992.

[9] T. A. Manteuffel, S. McCormick, J. Morel, S. Oliveira, and G. Yang, *A parallel version of a multigrid algorithm for isotropic transport equations*, SIAM J. Sci. Comput., 15 (1994), pp. 474–493.

[10] T. A. Manteuffel, S. McCormick, J. Morel, S. Oliveira, and G. Yang, *A fast multigrid algorithm for isotropic transport problems* I: *Pure scattering*, SIAM J. Sci. Comput., 16 (1995), pp. 601–635.

[11] T. A. Manteuffel, S. McCormick, J. Morel, and G. Yang, *A Fast Multigrid Algorithm for Isotropic Transport Problems* II: *With Absorption*, Los Alamos National Laboratories Report, LA-UR-94-2361, Los Alamos, NM, June, 1994.

[12] S. F. McCormick, *Multilevel Adaptive Methods for Partial Differential Equations*, SIAM, Philadelphia, PA, 1989.

[13] J. E. Morel and E. W. Larsen, *A multiple balance approach for differencing the S_N equations*, Nuclear Sci. Eng., 105 (1990), p. 1.

[14] J. E. Morel and T. A. Manteuffel, *An angular multigrid acceleration technique for the S_N equations with highly forward-peaked scattering*, Nuclear Sci. Eng., 107 (1991), p. 330.

[15] S. Oliveira, *Parallel Multigrid Methods for Transport Equations*, Ph.D. thesis, Department of Mathematics, University of Colorado at Denver, Denver, CO, 1993.

SIAM J. SCI. COMPUT.
Vol. 17, No. 6, pp. 1475–1490, November 1996

010

JACOBIAN-WEIGHTED ELLIPTIC GRID GENERATION*

PATRICK M. KNUPP†

Abstract. Variational grid generation techniques are used to derive and analyze a weighted elliptic grid generator that controls the Jacobian of the underlying transformation in a least-squares sense. The Euler–Lagrange equations for the area and volume generators are weighted forms of the well-known Laplace generator. Weights are restricted to the class of P-matrices to help achieve global invertibility of the map. Connecting the weights to the Jacobian of the map results in an intuitive means of controlling grid spacing, area, orthogonality, and grid-line directions. Examples are given on the unit square to demonstrate point attraction, local refinement, directional alignment, and adaption to a shock.

Key words. elliptic grid generation, adaptive grid

AMS subject classification. 65M50

1. Introduction. Elliptic grid generation, as originally proposed by Winslow [20] and subsequently developed by Thompson, Thames, and Mastin [18], entails solving a set of uncoupled Laplace equations for the logical coordinate variables subject to Dirichlet data describing the boundary of some domain. The so-called "boundary-fitted" coordinate system that results is used to compute finite difference or finite volume solutions to physical problems on geometrically complicated domains that involve partial differential equations (PDEs) such as the heat equation, convection-diffusion, and the Navier–Stokes equations without need for interpolation of the physical boundary conditions. The principle advantages of elliptic grid generation using the Laplace equation are that the resulting coordinate transformation is smooth and, in many cases, nonsingular. A major limitation of the Laplace generator is that there is no control over the grid that is generated (except for weak control via the boundary parameterization). Often the unique grid that is the solution to the Laplace system of equations is unsatisfactory for the physical problem at hand due to inappropriate spacing, overly large cells, or lack of orthogonality. A widely accepted approach to overcoming this limitation is to introduce user-specified weighting functions that generalize the Laplace equations to a system of Poisson equations [18]. It should come as no surprise that this does indeed provide qualitative control over the grid since the addition of any "source term" must surely modify the grid in some way. Because of this flexibility it is not unreasonable to ask that the weightings have a clear geometric meaning. By this criterion it is not clear that the weighting suggested in [18] is optimal. An alternate weighted form of the Laplace generator (referred to as the "Jacobian-weighted" generator) is proposed for which the weights have a clear geometric interpretation. As a bonus, the generator includes additional control capabilities beyond those usually claimed for weighted elliptic generators.

2. The variational principle. Let $n = 2$ or $n = 3$ be the dimension of the physical space in which grid generation is to be performed. Define a unit logical domain U_n as the Cartesian product of the unit interval n times: $U_n = \{(\xi_1, \dots, \xi_n) \mid 0 \le \xi_i \le 1\}$ for $i = 1, \dots, n$. Let $\Omega \subset R^n$ be a bounded, simply connected "physical" domain. We seek a smooth (at least $C^2(\Omega)$) one-to-one mapping $x_i = x_i(\xi_1, \dots, \xi_n)$ (or, in shorthand notation, $\mathbf{x} = \mathbf{x}(\xi)$) from U_n to Ω whose Jacobian matrix $\mathcal{J}$,

$$[\mathcal{J}]_{i,j} = \frac{\partial x_i}{\partial \xi_j}, \tag{1}$$

*Received by the editors December 12, 1994; accepted for publication (in revised form) June 22, 1995.

†Ecodynamics Research Associates, P.O. Box 9229, Albuquerque, NM 87119 (pknupp@math.unm.edu).

obeys certain requirements. The inverse mapping is given by $\xi_i = \xi_i(x_1, \dots, x_n)$, with inverse Jacobian matrix $\mathcal{J}^{-1}$ having the elements

$$[\mathcal{J}^{-1}]_{i,j} = \frac{\partial \xi_i}{\partial x_j} . \tag{2}$$

Our goal is to control the full Jacobian matrix of the map. To do so, let $\mathcal{S}$ be an $n \times n$ square, nonsingular weight matrix defined on Ω. A direct way to use this weight would be to solve the system $\mathcal{J}^{-1} = \mathcal{S}$, giving n^2, first-order equations

$$\frac{\partial \xi_i}{\partial x_j} = [\mathcal{S}]_{i,j} . \tag{3}$$

A major difficulty with this direct approach is that such a system of equations is overdetermined because an additional set of constraints

$$\frac{\partial [\, S\,]_{i,j}}{\partial x_k} = \frac{\partial [\, S\,]_{i,k}}{\partial x_j} \tag{4}$$

must be satisfied (these are derived from (3)). To fix this problem one should add the requirement that the weight matrix be the gradient of some continuous vector potential function. Such a vector potential would not be easy to construct as it essentially requires knowledge of the solution coordinates. Furthermore, even if such a potential were found, this system of first-order equations is not easily solved, particularly in view of the necessity of matching prescribed boundary conditions on $\partial\Omega$.

Given these considerations, it makes sense instead to perform a "least-squares" fit to the weight $\mathcal{S}$. Consistency conditions then need not be met and a vector potential is not required. The tools for such a least-squares approach are found in the method of variational grid generation [2], [15]. Following the former reference, one minimizes a functional of the form

$$I\,[\mathcal{J}^{-1}] = \int_{\Omega} G(\mathbf{x}, \mathcal{J}^{-1})\, d\mathbf{x} \tag{5}$$

subject to given boundary data, where $G \mid R^{n^2+n} \rightarrow R$ is an at least twice differentiable function of its arguments. The smoothness principle of Brackbill and Saltzman, which leads to the Laplace generator, is a critical example.

Prior to the present work, weighted variational principles have been formulated to control the elements $g_{i,j}$ of the metric tensor [15] or its inverse [2]. To improve upon this one must control the Jacobian matrix $\mathcal{J}$ (or its inverse). We thus formulate a variational principle that performs a least-squares fit of the inverse Jacobian matrix to the weight $\mathcal{S}$. The integrand in (5) that gives the Jacobian-weighted variational principle is

$$G = \mid \mathcal{J}^{-1} - \mathcal{S} \mid^2 , \tag{6}$$

where $\mid \mathcal{M} \mid^2 = \operatorname{tr}(\mathcal{M}^T \mathcal{M})$ is the square of the Frobenius norm of the matrix $\mathcal{M}$. Thus G is simply the sum of the squares of the differences $[\mathcal{J}^{-1} - \mathcal{S}]_{i,j}$. One could, of course, consider alternate matrix norms to measure the "closeness" of the two matrices. The Frobenius norm is presently favored since the resulting generator is then a weighted form of the widely accepted Laplace generator.

The stated variational principle compares favorably with Brackbill's weighted combination of the three functionals: smoothness, area, and orthogonality. As noted in [3, p. 39] there are problems with weighted combinations of functionals: one does not know a priori what weights to choose for the combination, one has no guarantee that the combination selected will result in a convex functional, and, finally, one has a scaling problem due to the dimensional heterogeneity of the functionals. These problems are avoided in the Jacobian-weighted functional because, by controlling the Jacobian of the map via a least-squares fit, only a single functional is needed to control grid spacing, area, orthogonality, and even the direction of the tangents.

The boundary data for the grid on $\partial\Omega$ is critical in determining the "goodness of fit," i.e., how close the value of I is to zero. If the boundary data and the weight matrix are incompatible, then the Jacobian of the minimizing grid will not agree with the weight—a least-squares fit will result. Even when the boundary data is compatible with the weight, $I = 0$ may not be attained unless the weight matrix is the gradient of a vector potential.

So far, no smoothness assumptions concerning the weight $\mathcal{S}$ have been mentioned. Since the goal is to generate smooth grids whose inverse Jacobian is potentially equal to the weight matrix, it is reasonable to require $\mathcal{S}$ to have some degree of smoothness. As will be seen in the next section, strong solutions to the Euler–Lagrange equations require $\mathcal{S} \in C^k(\Omega)$ with $k \geq 1$. When $k = 1$, the minimization should lead to mappings whose coordinates are nearly twice differentiable.

Since the integrand of the Jacobian-weighted functional is a convex function of the elements of $\mathcal{J}^{-1}$, it should be possible to show strong convexity of the functional using the techniques in [11]. In addition, the Legendre–Hadamard ellipticity condition is probably satisfied. If so, the generator is elliptic and a unique minimum of the functional exists. A rigorous analysis to show this has yet to be carried out.

3. The grid generation equations. Since the weighted variational principle (6) is a generalization of Brackbill and Saltzman's "smoothness" principle, one should not be surprised to learn that the Euler–Lagrange equations are a weighted form of the Laplace generator. Indeed, one finds that the coordinates of the inverse mapping satisfy a set of Poisson equations:

$$\nabla^2_{\mathbf{x}}\xi_i = \sum_{k=1}^{n} \frac{\partial S_{i,k}}{\partial x_k} \tag{7}$$

with $i = 1, \ldots, n$ and $S_{i,k}$ the elements of the weight matrix $\mathcal{S}$. To compare with standard Poisson grid generators, we explicitly give the equations for the planar form of the Jacobian-weighted generator:

$$\nabla^2_{\mathbf{x}}\xi = \frac{\partial S_{11}}{\partial x} + \frac{\partial S_{12}}{\partial y}, \tag{8}$$

$$\nabla^2_{\mathbf{x}}\eta = \frac{\partial S_{21}}{\partial x} + \frac{\partial S_{22}}{\partial y}. \tag{9}$$

We prefer to write (7) in the symmetric form

$$\operatorname{div}_{\mathbf{x}}(\mathcal{J}^{-1} - \mathcal{S}) = \mathbf{0} \tag{10}$$

because this makes it clear that a potential solution is $\mathcal{J}^{-1} = \mathcal{S}$, boundary conditions permitting. If $\mathcal{S}$ is any matrix whose divergence is zero, the system reduces to the Laplace system of equations.

Since, in general, the grid on Ω is not uniform it is not easy to numerically solve the Euler–Lagrange equations directly (but see the work of Hagmeijer [5] for an example of how this can be done). For numerical computation, it is convenient (and standard) to invert the equations so that the logical variables become the independent variables. The lengthy derivation of the inverted Laplace equations is seldom given since the result is the well-known Winslow equation

$$g_{22}\,\mathbf{x}_{\xi\xi} - 2\,g_{12}\,\mathbf{x}_{\xi\eta} + g_{11}\,\mathbf{x}_{\eta\eta} = \mathbf{0} \tag{11}$$

(as shorthand, we denote the left-hand side operator of the Winslow equations by $\mathcal{Q}_w\,\mathbf{x}$). As shown in [8, p. 154], the lengthy derivation of the Winslow equations can be condensed into just a few steps. That derivation is extended here for (10) in order to obtain inverted, nonsymmetric, Jacobian-weighted equations that can be conveniently solved numerically. Let $\mathcal{C} \equiv \sqrt{g}\,\mathcal{J}^{-T}$. Then

$$\begin{aligned}
\mathrm{div}_{\mathbf{x}}\mathcal{J}^{-1} &= \mathrm{div}_{\mathbf{x}}\mathcal{S}\,,\\
\mathrm{div}_{\xi}\,\mathcal{J}^{-1}\mathcal{C} &= \mathrm{div}_{\xi}\mathcal{S}\mathcal{C}\,,\\
\mathrm{div}_{\xi}\sqrt{g}\mathcal{G}^{-1} &= [\nabla_{\xi}\mathcal{S}]\mathcal{C}\,,\\
-\sqrt{g}\mathcal{J}\mathrm{div}_{\xi}\sqrt{g}\mathcal{G}^{-1} &= -\sqrt{g}\mathcal{J}[\nabla_{\xi}\mathcal{S}]\mathcal{C}\,,\\
g\,[\nabla_{\xi}\mathcal{J}]\,\mathcal{G}^{-1} &= -\sqrt{g}\,\mathcal{J}\,[\nabla_{\xi}\mathcal{S}]\mathcal{C}\,,\\
\mathcal{Q}_w\mathbf{x} &= -\sqrt{g}\,\mathcal{J}\,[\nabla_{\xi}\mathcal{S}]\mathcal{C}\,.
\end{aligned} \tag{12}$$

Explicitly, the inverted equations (12) in two dimensions are

$$g_{22}\,\mathbf{x}_{\xi\xi} - 2\,g_{12}\,\mathbf{x}_{\xi\eta} + g_{11}\,\mathbf{x}_{\eta\eta} = -\sqrt{g}\,\mathcal{J}\,\mathbf{R} \tag{13}$$

with right-hand side vector

$$\mathbf{R} = \begin{bmatrix} (S_{11})_{\xi}\,y_{\eta} - (S_{12})_{\xi}\,x_{\eta} - (S_{11})_{\eta}\,y_{\xi} + (S_{12})_{\eta}\,x_{\xi} \\ (S_{21})_{\xi}\,y_{\eta} - (S_{22})_{\xi}\,x_{\eta} - (S_{21})_{\eta}\,y_{\xi} + (S_{22})_{\eta}\,x_{\xi} \end{bmatrix}. \tag{14}$$

As shown in [8], one can form projections of these equations in physical space and also derive a symmetric inverted form. Since our numerical scheme is based on solving (13)–(14) these alternate forms are not given here.

4. Ellipticity. It is commonly accepted that the Winslow operator in (13) is elliptic. For completeness, we sketch a proof of this fact here using the ellipticity test given in [8, p. 127]. The grid equations have the form

$$\sum_{i,j=1}^{n} \mathcal{T}_{i,j}\mathbf{x}_{\xi_i\xi_j} = \mathbf{F} \tag{15}$$

with $\mathcal{T}_{i,j}$ being $n \times n$ coefficient matrices and $\mathbf{F}$ being a right-hand side comprised of lower-order terms. The following ellipticity test applies: the system is elliptic if one can find a positive constant c that is independent of $\omega = (\omega_1, \ldots, \omega_n)$ and $\mathbf{x}$ such that

$$\mathbf{det}\left(\sum_{i,j=1}^{n} \mathcal{T}_{i,j}\omega_i\omega_j\right) \geq c\mid\omega\mid^{2n}\ . \tag{16}$$

Evaluation of the left-hand side of this expression for $\mathcal{Q}_w\,\mathbf{x}$ converts the ellipticity requirement to

$$(\omega^T g\mathcal{G}^{-1}\omega)^n \geq c\,(\omega^T\omega)^n\,. \tag{17}$$

On the left-hand side is recognized the "elliptic norm" of ω under the symmetric, positive-definite matrix $g\,\mathcal{G}^{-1}$. A well-known property of such elliptic norms [6] is that

$$0 < \lambda_m\, \omega^T \omega \leq \omega^T g \mathcal{G}^{-1} \omega \leq \lambda_M\, \omega^T \omega, \tag{18}$$

where λ_m and λ_M are the minimum and maximum eigenvalues of $g\mathcal{G}^{-1}$, respectively. We thus choose $c = (\inf_{\mathbf{x}\in\Omega} \lambda_m)^n$ to satisfy the ellipticity condition (17).

5. Grid replication. It is clear that $\mathcal{J}^{-1} = \mathcal{S}$ is a solution to (10) (with no boundary conditions). It is perhaps not as clear that it is also a solution to the inverted equations (12), but this fact may be verified using the identity

$$\mathcal{Q}_w \mathbf{x} \equiv -\sqrt{g}\, \mathcal{J}\, [\, \nabla_\xi \mathcal{J}^{-1}\,]\, \mathcal{C}\,, \tag{19}$$

which holds for all parameterizations of the domain.

The identity (19) is easily proved beginning with another, more obvious identity:

$$[\, \nabla_\xi \mathcal{I}\,]\, \mathcal{J}^{-T} \equiv \mathbf{0}\,, \tag{20}$$

$$[\, \nabla_\xi\, (\mathcal{J}\mathcal{J}^{-1})\,]\, \mathcal{J}^{-T} \equiv \mathbf{0}\,, \tag{21}$$

$$\mathcal{J}\, [\, \nabla_\xi \mathcal{J}^{-1}\,]\, \mathcal{J}^{-T} + [\, \nabla_\xi \mathcal{J}\,] \mathcal{J}^{-1} \mathcal{J}^{-T} \equiv \mathbf{0}\,. \tag{22}$$

The result (19) directly follows.

Since $\mathcal{J}^{-1} = \mathcal{S}$ is a generic solution to the grid equations, it is clear that for any smooth mapping from U_n to Ω there exists a weight $\mathcal{S}$ that will replicate this mapping provided the proper boundary data is applied to the system of grid equations. This result holds for the continuum. If the grid equations are not discretized in just the right way (at this point it is not clear that such a way exists), truncation error will foil exact replication of most discrete grids. This truncation error effect on grid replication may become important if the desired coordinate system is highly stretched. To save space, an investigation as to whether or not one can discretize (13)–(14) to maintain replication is reserved for a future article.

6. Relationship to other generators. The fact that the present generator controls (in a least-squares sense) the Jacobian of the map means that it can be used to approximately align the coordinate system with a given set $\mathbf{v}_1, \ldots, \mathbf{v}_n$ of vector fields; examples for $n = 2$ are given in [9]. It is appropriate then to compare, in general terms, the Jacobian-weighted method with other generators which have this alignment capability. The variationally based alignment generator due to Giannakopoulos and Engel [4] minimizes a functional having the integrand (in two dimensions)

$$G = \mid \mathbf{v}_2^{\perp} \times \nabla_{\mathbf{x}} \xi \mid^2 + \mid \mathbf{v}_1^{\perp} \times \nabla_{\mathbf{x}} \eta \mid^2 \tag{23}$$

to achieve "directional control" (we have modified their original notation to coincide with ours). The geometric idea behind this principle is that the cross products are minimized when the vector $\mathbf{v}_2^{\perp}$ is parallel to $\nabla_{\mathbf{x}}\xi$ and the vector $\mathbf{v}_1^{\perp}$ is parallel to $\nabla_{\mathbf{x}}\eta$ (here $\mathbf{v}_i^{\perp}$ is a vector perpendicular to $\mathbf{v}_i$). In terms of covariant vectors, the functional is minimized when $\mathbf{v}_i$ is parallel to (i.e., aligned with) $\mathbf{x}_{\xi_i}$. Examples are given in [4] to show that this indeed works, at least in some cases. As noted in [8, p. 224], the integrand (23) may be rewritten in the form

$$G = \nabla_{\mathbf{x}}\xi \cdot (\mathbf{v}_2 \otimes \mathbf{v}_2) \nabla_{\mathbf{x}}\xi + \nabla_{\mathbf{x}}\eta \cdot (\mathbf{v}_1 \otimes \mathbf{v}_1) \nabla_{\mathbf{x}}\eta\,. \tag{24}$$

The matrices in this latter expression are vector outer products. Such product matrices are singular; thus it is not unreasonable to expect that the grid equations corresponding to this variational principle are nonelliptic.

Applying the method of §4, the ellipticity of the Giannakopoulos and Engel functional (for $n = 2$) is considered. Let $\mathcal{A} = \mathbf{v}_2 \otimes \mathbf{v}_2$ and $\mathcal{B} = \mathbf{v}_1 \otimes \mathbf{v}_1$. Define matrices $\mathcal{R} = \mathcal{J}^{-1}\mathcal{A}\mathcal{J}^{-T}$, $\mathcal{S} = \mathcal{J}^{-1}\mathcal{B}\mathcal{J}^{-T}$. The Euler–Lagrange equations corresponding to the functional in (24) are given in equations (11.120)–(11.121) of [8, p. 236]. The latter may be written in the form (15) where

$$\mathcal{T}_{i,j} = \operatorname{diag}([\mathcal{R}]_{i,j}, [\mathcal{S}]_{i,j})\, \mathcal{J}^{-1} \,. \tag{25}$$

Evaluation of the left-hand side of (16) for this functional gives

$$\frac{(\omega^T \mathcal{R}\, \omega)\, (\omega^T \mathcal{S}\, \omega)}{\sqrt{g}} \geq c(\omega^T \omega)^2 \tag{26}$$

as the ellipticity condition. Since $\mathcal{A}$ and $\mathcal{B}$ are singular, so are $\mathcal{R}$ and $\mathcal{S}$. If $\omega \neq 0$ is in the null space of either of the latter matrices, the left-hand side of the previous equation is zero. There does not exist a positive constant c, so the Giannakopoulos and Engel alignment functional is, at best, degenerate elliptic.

This analysis is supported by Brackbill's proposal to form a combination of the directional control functional (23) and the "smoothness" functional to improve the "regularity" of the equations [3]. A problem with such combinations is that they involve trade-offs between the various properties. One might expect that the more one weights the combination toward "smoothness," the less one will achieve directional control and vice versa.

We have little doubt that other functionals have been or may be devised which would achieve alignment (see, for example, [5]). We are not in a position to judge whether or not the presently proposed generator can align grids better than these other methods because to date it has not been possible to run the different algorithms on the same problem. It is possible that all of these generators do comparable jobs of grid alignment; however it seems that the Jacobian-weighted approach has several significant advantages. First, the generator reduces to the well-established Laplace generator in the unweighted case, so it has all the strengths (including ellipticity) of the latter generator, at least in the limiting case. Second, the proposed generator can be used to control other properties of the grid besides alignment (see §10), whereas the "directional control" generator is designed specifically for the alignment task.

What is the relationship of the present scheme to the Thompson–Thames–Mastin (TTM) elliptic generator [18]? Recall that the latter employs weight functions P and Q in the Poisson system

$$\nabla^2_{\mathbf{x}}\xi = P \,, \tag{27}$$

$$\nabla^2_{\mathbf{x}}\eta = Q \,. \tag{28}$$

In the original TTM generator, the weights P and Q have the complicated forms

$$\begin{aligned} P(\xi, \eta) = &- \sum_{m=1}^{M} a_m \frac{\xi - \xi_m}{\mid \xi - \xi_m \mid}\, e^{-c_m |\xi - \xi_m|} \\ &- \sum_{i=1}^{I} b_i \frac{\xi - \xi_i}{\mid \xi - \xi_i \mid}\, e^{-d_i\,[\,(\xi-\xi_i)^2 + (\eta-\eta_i)r\,]^{\frac{1}{2}}} \,, \end{aligned} \tag{29}$$

$$\begin{aligned} Q(\xi, \eta) = &- \sum_{m=1}^{M} a_m \frac{\eta - \eta_m}{\mid \eta - \eta_m \mid}\, e^{-c_m |\eta - \eta_m|} \\ &- \sum_{i=1}^{I} b_i \frac{\eta - \eta_i}{\mid \eta - \eta_i \mid}\, e^{-d_i\,[\,(\xi-\xi_i)^2 + (\eta-\eta_i)r\,]^{\frac{1}{2}}} \end{aligned} \tag{30}$$

(see [18] for details of how this is supposed to work). By specifying the various parameters a_m, c_m, b_i, and d_i, it is claimed that one can attract or repel grid lines to M points and I lines within the physical domain. Examples show that this qualitatively works, at least for a small number of attraction points/lines. The forms (29)–(30) of the weights provide the user with neither orthogonality nor grid alignment controls.

Warsi [19] replaced the original P and Q weights defined in (29)–(30) by $g^{11}\,P$ and $g^{22}\,Q$ to improve the numerical behavior of the generator—no additional control over the grid was claimed. Thomas and Middlecoff [16] replaced the P and Q in the Warsi modification with $P = \phi(\xi, \eta)$ and $Q = \psi(\xi, \eta)$ and abandoned formulas (29)–(30) for the weights in favor of performing interpolations from the boundary data. In Anderson [1], the new control functions ϕ and ψ are constructed to satisfy

$$\phi = -\frac{\partial}{\partial \xi} \ln \sqrt{g_{11}}\,, \tag{31}$$

$$\psi = -\frac{\partial}{\partial \eta} \ln \sqrt{g_{22}} \tag{32}$$

so that the weights potentially control the normalized rate of change of grid spacing in the two logical directions (the derivation of these relationships is not strictly correct because it is assumed that the grid is orthogonal). Nonetheless, the resulting grid generator has proved useful in numerous instances, particularly if physical space weightings are used (as in [14]).

The Jacobian-weighted generator defines weights P and Q by

$$P = \frac{\partial}{\partial x} S_{11} + \frac{\partial}{\partial y} S_{12}\,, \tag{33}$$

$$Q = \frac{\partial}{\partial x} S_{21} + \frac{\partial}{\partial y} S_{22}, \tag{34}$$

where the $S_{i,j}$ are the elements of a weight matrix $\mathcal{S}$ which controls the elements of the inverse Jacobian matrix of the transformation. Superficially then, the generator is a special case of the P-Q generator of (27)–(28). On a deeper level, however, it cannot be considered a special case because the traditional P-Q generator is derived from a different variational principle. The integrand in the variational principle for the P-Q generator (with physical space weightings $P = P(\mathbf{x})$ and $Q = Q(\mathbf{x})$) is

$$G = \frac{1}{2} \mid \mathcal{J}^{-1} \mid^2 + P\,\xi + Q\,\eta\,, \tag{35}$$

which has no obvious geometric meaning.

It is not claimed that the present Jacobian-weighted method always does a better job of controlling the grid than does the P-Q method. In fact, any grid generated by one of the methods can be generated by the other provided the proper weight is selected. However, the Jacobian-weighted method should ultimately prove preferable because of the clear meaning of its control functions and because it can potentially control grid orthogonality and grid alignment, in addition to all of the properties controlled by P-Q.

7. Nonsingular mappings. Boundary-fitted coordinates are often used in computational fluid dynamics to solve a system of PDEs that models some physical phenomenon such as fluid flow or heat transfer. To apply these coordinates one must transform the physical equations to general coordinates. To avoid a change of type of the governing equations requires the mapping to be locally invertible. It is therefore desirable to have a guarantee that the mapping produced by a grid generator will be locally invertible, i.e., nonsingular. Such a guarantee has been constructed for the 2D Winslow generator $\mathcal{Q}_w\mathbf{x} = \mathbf{0}$ [12]. The proof of this guarantee hinges on the maximum principle for Laplace's equation in the plane. Even this guarantee does not always avoid trouble when the equations are discretized: the results in §5.4.3 of [8] suggest that truncation error effects can sometimes prevent the discrete grids from being nonsingular. Examples have also been constructed to show that the guarantee that holds for the 2D Laplace equations does not extend to the 3D case (see [10] for a discussion of the extension of Rado's theorem to three-dimensions).

A more serious limitation is that, in practice, the weighted P-Q forms of the Laplace generator are often used. This case is not covered by the guarantee that holds for the unweighted equations. As observed in [14], there can be no guarantee that the P-Q generator will not produce a folded grid because given any smooth but folded mapping one can compute P and Q directly from the Laplacian of the map. If these weights are used in the Poisson generator, then the folded mapping will be reproduced. In contrast to the unweighted generator, it is easy to produce folded maps with the P and Q generator (see Project 5.5.1 in [8, p. 109]).

From these observations it seems that one cannot hope for an invertibility guarantee for the present Jacobian-weighted elliptic generator. However, in the next section we will suggest a condition upon the weight matrix that, although not a guarantee, will provide a guideline for constructing weights that have a reasonable chance of producing a nonsingular grid.

8. Global univalence. A sufficient (but not necessary) condition to achieve *local* invertibility of a mapping is the requirement that the Jacobian matrix be invertible; equivalently, one needs $\sqrt{g} \neq 0$. There has been considerable work [13] in determining what are sufficient conditions to guarantee that a map is *globally* invertible.

We focus on one of several results that is known on this subject because it seems to fit naturally with the grid generation problem. We need the following definition.

DEFINITION. *An $n \times n$ real matrix $\mathcal{A}$ is a P-matrix if every principle minor of $\mathcal{A}$ is positive.*

Recall that the principle minors of a square matrix are the determinants of the square submatrices of the matrix. For example, if $n = 2$, the principle minors of $\mathcal{A}$ are a_{11}, a_{22}, and $\det \mathcal{A}$. If $n = 3$, the principle minors are a_{11}, a_{22}, a_{33}, $a_{11}a_{22} - a_{21}a_{12}$, $a_{11}a_{33} - a_{31}a_{13}$, $a_{22}a_{33} - a_{32}a_{23}$, and $\det \mathcal{A}$. It is shown in [13] that if $\mathcal{A}$ is nonsingular and a P-matrix, then $\mathcal{A}^{-1}$ is a P-matrix. We base our argument in this section upon the following fundamental result of Gale, Nikaido, and Inada.

FUNDAMENTAL GLOBAL UNIVALENCE THEOREM. *Let $F : U \subset R^n \rightarrow R^n$ be a differentiable mapping, where U is the rectangular region $U = \{\mathbf{x} : \mathbf{x} \in R^n \mid a_i \leq x_i \leq b_i\}$. If the Jacobian matrix of F at $\mathbf{x}$, $\mathcal{J}(\mathbf{x})$, is a P-matrix for every $\mathbf{x} \in U$, then F is globally univalent in U.*

In general, the Jacobian matrix $\mathcal{J}$ is not a P-matrix, so the mappings produced for arbitrary weights $\mathcal{S}$ cannot be expected to be globally invertible. The univalence theorem suggests that it is reasonable to require that the weight matrix $\mathcal{S}$ in the Jacobian-weighted elliptic grid generator be a P-matrix. If $\mathcal{J}^{-1} = \mathcal{S}$ is achieved, then $\mathcal{J} = \mathcal{S}^{-1}$ is a P-matrix and the map will be globally univalent. Even if $\mathcal{J}^{-1} = \mathcal{S}$ is not achieved, it is possible that $\mathcal{J}^{-1}$ will still be a P-matrix because matrices "near" $\mathcal{S}$ are still P-matrices. P-matrix weights do not guarantee global invertibility of the mapping but can serve as a guide to help construct useful weights. A subset of the P-matrices is the set of positive, quasi-definite matrices.

DEFINITION. *An $n \times n$ (not necessarily symmetric) real matrix $\mathcal{A}$ is a positive* quasi-definite *matrix if $\mathbf{x}^T \mathcal{A}\mathbf{x} > 0$ for all $\mathbf{x} \neq 0$.*

Since such matrices are more strongly constrained than are P-matrices, requiring the weight $\mathcal{S}$ to be positive, quasi definite would seem to help achieve global invertibility. The P-matrix and quasi-definite criteria on the weight will be refined in the next section.

9. The form of the weight matrix. The weight function $\mathcal{S}$ controls the inverse Jacobian matrix of the transformation. A crucial question asks how this weight should be constructed. Consider the planar case $\Omega \subset R^2$ first.

In the general planar case, one can consider alignment of the two tangent vectors $\mathbf{x}_\xi$ and $\mathbf{x}_\eta$ with two unit vectors $\mathbf{u}_j^T = (u_j^1, u_j^2)$, $j = 1, 2$. Let $\mathcal{U}$ be a 2×2 matrix with elements $U_{i,j} = u_j^i$. It is sometimes useful to express the unit vectors in terms of "direction cosines" as

$$\mathcal{U} = \begin{pmatrix} \cos\theta_1 & \cos\theta_2 \\ \sin\theta_1 & \sin\theta_2 \end{pmatrix}, \tag{36}$$

where $-\pi < \theta_j \leq \pi$ forms the angle between the jth tangent vector and the x-axis. To establish the length scale of the map, let $\mathcal{L} = \text{diag}(\ell_1, \ell_2)$ with $\ell_j > 0$. The weight matrix is written as the inverse product

$$\mathcal{S} = (\mathcal{U}\mathcal{L})^{-1}. \tag{37}$$

For the special case of a single direction $\mathbf{u}_1$, given it makes sense to require grid orthogonality, let $\theta_1 = \theta$ be the angle between the x-axis and the known unit vector $\mathbf{u}_1$. Let $\theta_2 = \theta + \pi/2$ in order that the unit vector $\mathbf{u}_2$ be orthogonal to $\mathbf{u}_1$. This gives

$$\mathcal{U} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}. \tag{38}$$

If the solution $\mathcal{J} = \mathcal{S}^{-1}$ is attained, then the following relations hold:

$$x_\xi = \ell_1 \cos\theta, \tag{39}$$

$$y_\xi = \ell_1 \sin\theta, \tag{40}$$

$$-x_\eta = \ell_2 \sin\theta, \tag{41}$$

$$y_\eta = \ell_2 \cos\theta. \tag{42}$$

These in turn imply

$$\ell_1 = \sqrt{g_{11}}, \tag{43}$$

$$\ell_2 = \sqrt{g_{22}}, \tag{44}$$

and

$$g_{12} = 0, \tag{45}$$

$$\sqrt{g} = \ell_1 \ell_2, \tag{46}$$

$$\sqrt{g_{22}/g_{11}} = \ell_2/\ell_1, \tag{47}$$

showing that this construction of the weight potentially gives an orthogonal grid with cell areas given by the product of the length-control functions, cell-aspect ratios given by the ratio of the length-control functions, and whose tangent $\mathbf{x}_\xi$ makes an angle θ with the x-axis.

To make use of the concept of P-matrices for the weight matrix requires further discussion. Consider the mapping $x = \eta$, $y = -\xi$ on the unit square. The Jacobian matrix for this map is

$$\mathcal{J} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \tag{48}$$

and the determinant is $\sqrt{g} = 1$. The map is thus nonsingular, yet the Jacobian is not a P-matrix because two of the minors are zero. The global univalence theorem is not refuted by this example because the requirement that the Jacobian be a P-matrix is a sufficient, not necessary, condition. Nevertheless, it is disturbing that the Jacobian for this simple map does not give a P-matrix.

To fix this problem we refine our notion of how P-matrices should be applied to grid generation. Instead of requiring that the Jacobian of the map be a P-matrix we shall require that there exists a finite set of rigid body rotations (given by the $n \times n$ matrix $\mathcal{R}$) such that the product $\mathcal{R}\,\mathcal{J}$ is a P-matrix. Intuitively, this means that we can rotate the local tangent vectors in such a way as to obtain a P-matrix. In the above example, if one lets $\phi = \pi/2$ and

$$\mathcal{R} = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix}, \tag{49}$$

then $\mathcal{R}\,\mathcal{J} = \mathcal{I}$, which is obviously a P-matrix.

We next determine what restrictions must be put on the weight $\mathcal{S}$ in (37) to ensure that $\mathcal{R}\,\mathcal{S}^{-1}$ is a P-matrix. Since

$$\mathcal{R}\,\mathcal{S}^{-1} = \begin{pmatrix} \ell_1 \cos(\theta_1 + \phi) & \ell_2 \cos(\theta_2 + \phi) \\ \ell_1 \sin(\theta_1 + \phi) & \ell_2 \sin(\theta_2 + \phi) \end{pmatrix} \tag{50}$$

one finds that sufficient conditions for the product to be a P-matrix are that $\ell_1 > 0$, $\ell_2 > 0$, and $0 < \theta_2 - \theta_1 < \pi$ (this can be seen by taking $\phi = -\theta_1$). Geometrically this means that the length scales are positive and the unit vectors are properly oriented (positive cross product). Restricting the weight matrix to a positive, quasi-definite matrix requires that $\theta_2 = \theta_1 + \pi/2$; i.e., an orthogonal grid is demanded.

In this construction of the weight matrix we are to supply three controls, ℓ_1, ℓ_2, and θ, that are functions of $\mathbf{x}$ which prescribe the two length scales and the direction of the tangents. Precise control (i.e., actually getting a grid for which $\mathcal{J}^{-1} = \mathcal{S}$) may not be achieved due to the fact that the elliptic equations produce solutions that are least-squares fits to the conditions we are attempting to impose. Lack of precise control can be helpful because one may have only a rough idea of values for these control functions. The advantage of the construction just given is not that it makes figuring out the weight function easy (it is not), but rather that the weight functions that must be defined have clear geometric meanings.

In three-dimensions, let ℓ_1, ℓ_2, and ℓ_3 be given length-control functions and $\mathcal{L} = \mathrm{diag}(\ell_1, \ell_2, \ell_3)$. Three unit vectors $\mathbf{u}_j$, $j = 1, 2, 3$ are given in terms of the angles ϕ_j and θ_j. The polar angle ϕ_j lies in the range $0 \le \phi_j \le \pi$ and gives the angle between the unit vector and the z-axis. The azimuthal angle θ_j lies in the range $0 \le \theta_j < 2\pi$ and gives the angle between the projection of the unit vector onto the x-y plane and the x-axis. The matrix $\mathcal{U}$ of components of the unit vectors has entries

$$\mathbf{u}_j = \begin{bmatrix} \cos\theta_j \, \sin\phi_j \\ \sin\theta_j \, \sin\phi_j \\ \cos\phi_j \end{bmatrix} \tag{51}$$

and the weight $\mathcal{S}$ is again formed as in (37). It is not clear what the useful special cases of this general setup in three-dimensions are, so no further assumptions are made at this time. It is expected that sufficient conditions for the 3D construction to be a P-matrix will turn out to be positive length-control functions and a well-oriented vector triple.

10. Examples on the unit square. Methods for discretizing and numerically solving grid generation equations of the type found in this paper are well known; examples may be found in [8]. It is well beyond the scope of this paper to consider all the ways in which the Jacobian-weighted generator may be used. The purpose in this section is to show that the control functions described in §9 indeed provide the control that is claimed and that reasonably good grids can be generated with minimal effort. The simple case in which the physical domain is the unit square seems sufficient for this purpose and has the advantage that the weight S can be given in terms of an analytic function of the physical space variables. Nonanalytic weights raise interpolation questions which cannot be addressed here due to lack of space.

A few preliminary remarks are in order before concrete examples are given. First, given a uniform parameterization of the boundary of the unit square, the natural transformation on the square is $x(\xi, \eta) = \xi$ and $y(\xi, \eta) = \eta$. The Jacobian matrix for this transformation is simply the identity. Uniform boundary data and a weight matrix $\ell_1 = \ell_2 = 1, \theta = 0$ gives the natural transformation because the gradient of this particular weight is zero and the uniform grid is a solution to the Winslow equations. Second, if a nonuniform parameterization of the boundary is given, then the identity weight will not reproduce the grid based on the natural uniform transformation because the weight is inconsistent with the boundary data. The solution to the Winslow equations with nonuniform boundary data thus produces a least-squares fit to a map having the identity as the Jacobian. Third, if the boundary parameterization is achieved by an exponential stretch of the logical variable, as in

$$x_B(\xi) = \frac{(1 - e^{\lambda \xi})}{(1 - e^{\lambda})}, \tag{52}$$

$$x_T(\xi) = x_B(\xi) \tag{53}$$

(with $\lambda \neq 0$), then the exponential map $x(\xi, \eta) = x_B(\xi)$, $y(\xi, \eta) = \eta$ will be produced by a weight matrix having a *linear* variation in x because then

$$x_\xi = \frac{\lambda}{1 - e^{\lambda}} [(1 - e^{\lambda}) x - 1]. \tag{54}$$

Example 1. *Local refinement.* Sometimes it is desired to refine a grid on a local subregion or patch to achieve greater accuracy in a part of the domain having a lot of solution structure. Often this is done by overlaying the main grid with a second grid having smaller cells on the subregion and performing an interpolation of the solution between the main grid and the patch grid. As a cheap alternative, the Jacobian-weighted generator can approximate patching by using a single grid to eliminate the need for interpolation.

As an example, let a patch $P \subset \Omega$ be the union of two rectangles: $R_1 = [1/4, 1/2] \times [1/2, 3/4]$ and $R_2 = [1/2, 3/4] \times [5/8, 3/4]$. The weight matrix control functions are $\theta = 0$ and $\ell_1 = \ell_2 = 1/2$ on P and unity otherwise (notice that we have violated our assumption that the weight is C^1 in this example). Nonetheless, Figure 1 shows that these simple controls permit the resulting elliptic grid to approximate subgrid patching. The transition zone between the patches could perhaps be improved by introducing a smooth transition function between the length scales on Ω and P (this would also improve the convergence rate of the Picard iteration for the solution). It would, of course, take considerably more work to define smooth control functions and the results may not justify the additional effort.

The main point in this example (and the ones below) is not that the grids generated are all that much better than those generated by other means, but merely that it is relatively easy to devise the control functions in the Jacobian-weighted method due to the clear meaning of the controls.

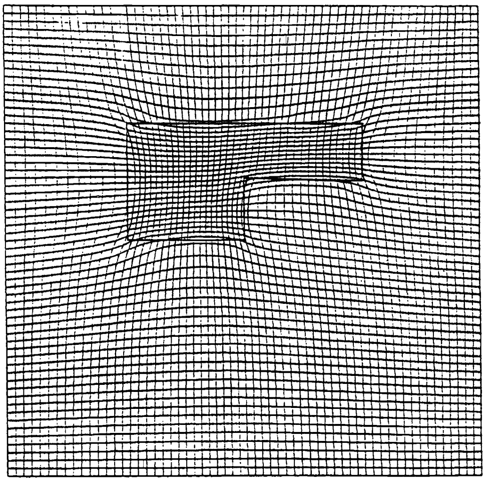


FIG. 1. *Local refinement.*

Example 2. *Point attraction.* In this example it is shown that weights can be devised to attract grid nodes to multiple points in the physical domain. Suppose the grid nodes are to be attracted to N distinct points (x_n, y_n), $n = 1, \ldots, N$ in the domain. The horizontal length-control function in this example takes the form

$$\ell_1(x, y) = 1 - \sum_{n=1}^{N} a_n B(r_n, R_n), \tag{55}$$

where B is a smooth "bump" function

$$B(r, R) = \begin{cases} 1 - 3(r/R)^2 + 2(r/R)^3, & r/R < 1, \\ 0 & \text{else} \end{cases} \tag{56}$$

chosen for the following properties: $B(0, R) = 1$, $B(R, R) = 0$, $B_r(0, R) = 0$, and $B_r(R, R) = 0$. Here r_n is the distance between the point (x, y) and the point (x_n, y_n). The parameters $0 < R_n < 1$ are "radii of attraction" and the parameters $0 < a_n < 1$ are "attraction strengths." The radii R_n are chosen so that the support of the "bump" functions do not overlap. The other control functions are $\ell_2 = \ell_1$ and $\theta = 0$. In Figure 2 we show results for the three attraction points $(x_1, y_1) = (1/4, 3/4)$, $(x_2, y_2) = (3/8, 1/4)$, and $(x_3, y_3) = (3/4, 5/8)$. The attraction radii were set to $R_n = 0.10$ and the attraction strengths were set to $a_n = 0.75$. Although the grid produced in this example is reasonably good (except for a lack of orthogonality in part of the domain), it is all too easy to produce poor grids or cause the Picard iteration to fail to converge by selecting extreme values of the parameters R_n and a_n. This will be exacerbated if the location of the attraction points is particularly demanding of a structured grid. If two or more of the attraction points lie close to one another, the separate attraction points will not be individually resolved unless more cells are added to the grid.

Example 3. *Feature adaptivity.* This example demonstrates that the alignment generator can be used as a feature-adaptive generator in much the same way as many other weighted grid generators. The example is based on a "solution" function borrowed from [7]:

$$f(x, y) = \tanh R\,\{r^2 - (x - x_c)^2 - (y - y_c)^2\}. \tag{57}$$

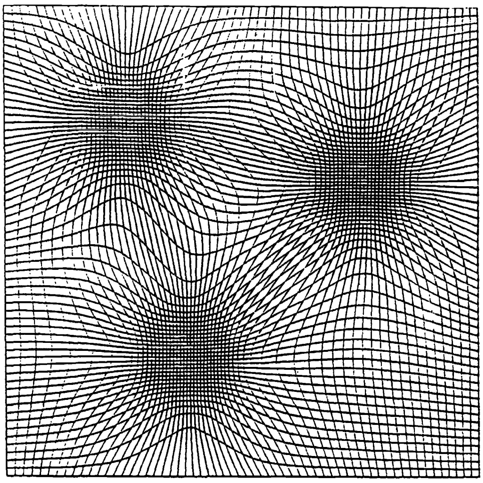


FIG. 2. *Point attraction.*

This function forms a circular "hill" with radius r centered about the point $x = x_c$, $y = y_c$ (in this example, $r = 1/4$, $x_c = 1/2$, and $y_c = 1/2$). Large values of R create a steep-sided hill. As is commonly done with adaptive generators, we wish to produce a grid with small cells at the points in physical space where $\mid \nabla_{\mathbf{x}} f \mid$ is large. To achieve this we set

$$\ell_1 = \frac{1}{(1 + a \mid \nabla_{\mathbf{x}} f \mid)}, \tag{58}$$

$$\ell_2 = \ell_1, \tag{59}$$

$$\theta = 0, \tag{60}$$

where a is a positive scale parameter. Four grids are shown in Figure 3: the first, Figure 3a, directly compares the Jacobian-weighted result for $R = 20$, $a = 0.2$ with Figure 3.5 in [7]. The grid in Figure 3a does not have the pinched cells found in Figure 3.5 in [7] because the present method is not based on 1D adaptions. Figure 3b shows the results of the Jacobian-weighted generator for a steeper hill ($R = 100$, $a = 0.05$). Figure 3c shows what happens for $R = 100$ when "a" in (58) is increased to $a = 0.10$—the grid has a cell of zero area (at least it did not fold over). This problem can be fixed by increasing the problem resolution from 40 × 40 to 60 × 60 cells. The result ($R = 100$ and $a = 0.10$) is given in Figure 3d.

The usual problems of feature-adaptive grid generation remain: determining the best choice for the parameter a. If a is too small, the grid does not respond to the gradient while if a is too large, the grid overresponds, resulting in a bad grid or even a folded grid (if one is lucky, the numerical iteration will fail to converge instead of producing a folded grid). What is the proper choice for the form of the control function (for example, why not use

$$\ell_1 = \frac{1}{\sqrt{1 + a^2 \mid \nabla_{\mathbf{x}} f \mid^2}} \tag{61}$$

instead of that defined in (58))?

Presently there is not enough evidence to claim that the Jacobian-weighted grid generator always gives better grids than other adaptive methods. It seems inescapable that a poor choice of parameters in the control functions of any method will produce a poor grid no matter how good the basic generator is. Thus the Jacobian-weighted grid generator (like all other known

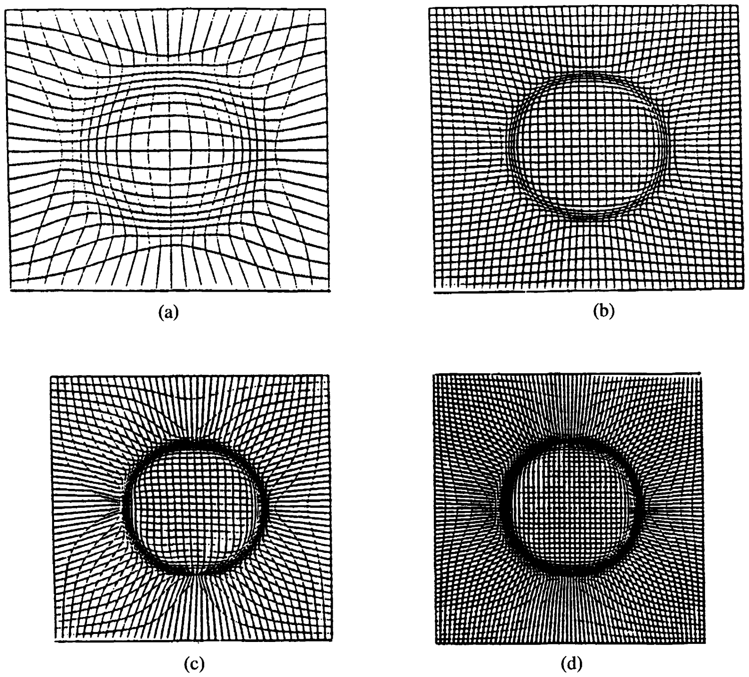


FIG. 3. *Gradient-weighted adaptivity:* (a) *Gentle hill.* $R = 20$, $a = 0.2$. (b) *Steep hill.* $R = 100$, $a = 0.05$. (c) *Steep hill.* $R = 100$, $a = 0.10$. (d) *Increased resolution.*

generators) cannot be considered an automatic generator, i.e., one which requires no user intervention. Instead, it is best if used in an interactive mode wherein the user can try a set of control functions and parameters, look at the grid produced, and adjust the parameters accordingly. From this viewpoint, the Jacobian-weighted generator seems attractive because at least the meaning of the weights is clear.

Example 4. *Adaption to a curved "shock".* This example also uses the "hill" function (57) but with $r = 5/4$, $x_c = 3/2$, and $y_c = 0$. With this choice the center of the "hill" lies outside the physical domain and the locus of points of maximum gradient traverses the domain in a circular arc. The goal is to create small cells near this arc and to align the horizontal tangent $\mathbf{x}_\xi$ so that it orthogonally traverses the arc. To do so, an annular sector of points (x, y) near the "shock" is identified by the relation $\mid r - \rho \mid < \epsilon$ where $\rho^2 = (x - x_c)^2 + (y - y_c)^2$. The parameter ϵ is the width of the annulus (taken to be 0.1 in this example). If the point (x, y) lies within this band, then the angle control function is taken to be

$$\theta = \arctan\left(\frac{y - y_c}{x - x_c}\right). \tag{62}$$

If the point (x, y) lies outside the annulus, then $\theta = 0$. Since the top boundary of the domain is not orthogonal to the "shock" we introduce a blending function

$$u(x) = \frac{(e^{\lambda x} - e^{\lambda})}{(1 - e^{\lambda})} \tag{63}$$

and multiply θ in (62) by u to reduce the amount of angle adaption near the top boundary ($\lambda = 2$ in the present example). The length-control functions are the same as in the previous example. The resulting grid shown in Figure 4 closely approximates the desired properties.

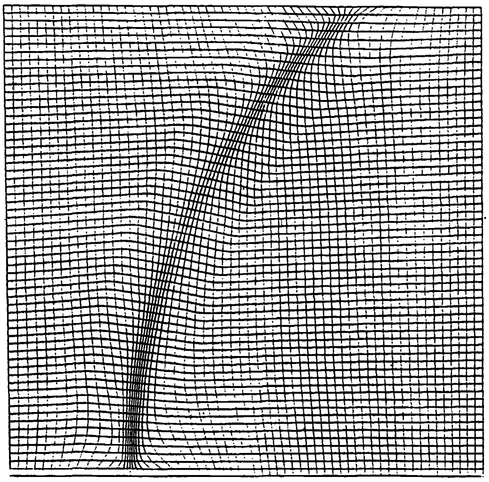


FIG. 4. *Adaption to a curved "shock."*

11. Summary. The "smoothness" variational principle of Brackbill and Saltzman has been generalized by performing a least-squares fit of the inverse Jacobian matrix to a weight matrix in order to control the Jacobian matrix of a mapping from a uniform logical domain to a bounded, simply connected domain in physical space. By specifying the Jacobian one can, in principle, control cell lengths, areas, orthogonality, and the direction of the tangent vectors, thus obviating the need for combining separate functionals for these properties into a single, possibly nonconvex, functional having mixed dimensional characteristics. The least-squares principle leads to "Jacobian-weighted" grid generation equations that are elliptic and constitute a generalization of the Laplace generator. The inverted equations are a weighted form of the well-known Winslow equations. Given any continuum transformation, the weight matrix that replicates the transformation is merely the inverse Jacobian matrix of the transformation. The Jacobian-weighted generator has a different variational principle than does the widely used "Poisson generator" and thus cannot be considered a special case of the latter. It is suggested that restricting the weight matrix to be a P-matrix will help give globally invertible mappings. Length-scale and directional control functions further define the weight matrix. Four examples were given to show that it is relatively easy to construct control functions to achieve local refinement, point attraction, feature adaptivity, and directional control.

The control functions of the Jacobian-weighted generator have clear geometric meanings and thus seem preferable to other weighted elliptic grid generators. Nevertheless, the method retains a number of the problems of previous elliptic methods such as lack of automation, lack of precision, and relative slowness due to the fact that a system of nonlinear PDEs must be solved. Further work should at least partially address these issues.

Many open issues regarding this generator remain. A partial list of such issues includes devising a discretization that will permit exact replication, designing fast, robust solvers for the discretized equations, testing of the method in three-dimensions, testing the method as an adaptive generator, finding efficient ways of constructing smooth, effective weight matrices, finding ways to ensure compatibility of the boundary data with the weight, parallelization of the algorithm, extension to a surface grid generation algorithm, determining fast and accurate interpolation methods when the weight is known only at discrete points, and determining if the generalized principle

$$G = w(\mathbf{x}) \mid \mathcal{J}^{-1} - \mathcal{S} \mid^2 \tag{64}$$

with $w > 0$ is of any use in forcing more precision at particular points in the domain. Extension of (13) to the case of surface grid generation will result in equations similar in form to those given in [17]. The left-hand side of the equation would remain the same while the right-hand side would be generalized to include a surface curvature term.

Acknowledgments. The author wishes to thank G. Liao, N. Robidoux, and S. Steinberg.

REFERENCES

[1] D. A. Anderson, *Equidistribution schemes, Poisson generators, and adaptive grids*, Appl. Math. Comput., 24 (1987), pp. 211–227.

[2] J. U. Brackbill and J. S. Saltzman, *Adaptive zoning for singular problems in two dimensions*, J. Comput. Phys., 46 (1982), pp. 342–368.

[3] J. U. Brackbill, *An adaptive grid with directional control*, J. Comput. Phys., 108 (1993), pp. 38–50.

[4] A. E. Giannakopoulos and A. J. Engel, *Directional control in grid generation*, J. Comput. Phys., 74 (1988), pp. 422–439.

[5] R. Hagmeijer, *Grid adaption based on modified anisotropic diffusion equations*, J. Comput. Phys., 115 (1994), pp. 169–183.

[6] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, 1985.

[7] W. Huang and D. Sloan, *A simple adaptive grid method in two dimensions*, SIAM J. Sci. Comput., 15 (1994), pp. 776–797.

[8] P. Knupp and S. Steinberg, *The Fundamentals of Grid Generation*, CRC Press, Boca Raton, FL, 1993.

[9] P. Knupp, *Mesh generation using vector-fields*, J. Comput. Phys., 119 (1995), pp. 142–148.

[10] G. Liao, *On harmonic maps*, in Mathematical Aspects of Numerical Grid Generation, J. E. Castillo, ed., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1991, pp. 123–130.

[11] ———, *Variational approach to grid generation*, Numer. Methods Partial Differential Equations, 8 (1992), pp. 143–147.

[12] C. W. Mastin and J. F. Thompson, *Elliptic systems and numerical transformations*, J. Math. Anal. Appl., 62 (1978), pp. 52–62.

[13] T. Pathasarathy, *On Global Univalence Theorems*, Lecture Notes in Math. 977, Springer-Verlag, New York, 1983.

[14] P. J. Roache, K. Salari, and S. Steinberg, *Hybrid adaptive Poisson grid generation and grid smoothness*, Comm. Appl. Numer. Methods, 7 (1991), pp. 345–354.

[15] S. Steinberg and P. Roache, *Variational grid generation*, Numer. Methods Partial Differential Equations, 2 (1986), pp. 71–96.

[16] P. Thomas and J. Middlecoff, *Direct control of the grid point distribution in meshes generated by elliptic equations*, AIAA J., 18 (1980), pp. 652–656.

[17] P. D. Thomas, *Composite three-dimensional grids generated by elliptic systems*, AIAA J., 20 (1982), pp. 1195–1202.

[18] J. F. Thompson, F. C. Thames, and C. W. Mastin, *Automatic numerical generation of body-fitted coordinates*, J. Comput. Phys., 15 (1974), pp. 299–319.

[19] Z. U. A. Warsi, *Basic differential models for coordinate generation*, in Numerical Grid Generation, J. F. Thompson, ed., North-Holland, New York, 1982, pp. 41–78.

[20] A. Winslow, *Numerical solution of the quasilinear Poisson equations in a nonuniform triangle mesh*, J. Comput. Phys., 2 (1967), pp. 149–172.

SIAM J. SCI. COMPUT.
Vol. 17, No. 6, pp. 1491–1508, November 1996

011

COMPOSITE STEP PRODUCT METHODS FOR SOLVING NONSYMMETRIC LINEAR SYSTEMS*

TONY F. CHAN† AND TEDD SZETO†

Abstract. First proposed in [*Numer. Math.*, 66 (1993), pp. 295–319, *Numer. Algorithms*, 7 (1994), pp. 1–16] by Bank and Chan, the composite step (CS) method is a technique for curing the pivot breakdown (one of two possible breakdowns) in the biconjugate gradient (BCG) algorithm by skipping over steps in which the iterate is not defined. We show how to extend this method to cure the same breakdown inherited in product methods such as conjugate gradients squared (CGS) [*SIAM J. Sci. Statist. Comput.*, 10 (1989), pp. 36–52], Bi-CGSTAB [*SIAM J. Sci. Statist. Comput.*, 13 (1992), pp. 631–644], Bi-CGSTAB2 [*Variants of BiCGSTAB for Matrices with Complex Spectrum*, IPS Research Report No. 91-14, ETH Zürich], which are derived from a product of the BCG polynomial with another polynomial of the same degree. New methods introduced in this paper are CS-CGSTAB (composite step applied to Bi-CGSTAB) and a more stable variant of this, CS-CGSTAB2, which handles a possible additional breakdown problem due to the Bi-CGSTAB process in the case where A is skew symmetric. CS-CGSTAB2 can be viewed as an improvement over the Bi-CGSTAB(l) algorithm (Sleijpen and Fokkema [*ETNA*, 1 (1993), pp. 11–32]) with $l = 2$ in that although Bi-CGSTAB(2) is designed to cure the skew-symmetric breakdown, it does not handle pivot breakdown as CS-CGSTAB2 does. The new methods require only a minor modification to the existing product methods. Moreover, the sizes of the steps taken in our methods can vary as opposed to fixed step methods like Bi-CGSTAB(l) and Bi-CGSTAB2. Our strategy for deciding whether to skip a step does not involve any machine dependent parameters and is designed to skip near breakdowns as well as produce smoother iterates. Numerical experiments show that the new methods do produce improved performance over those without composite step on practical problems. Furthermore, we extend the "best approximation" result in [*Numer. Algorithms*, 7 (1994), pp. 1–16] to obtain convergence proofs for CGS and Bi-CGSTAB and their composite step stabilized versions.

Key words. Lanczos method, biconjugate gradient method, breakdowns, product methods, Bi-CGSTAB, composite step

AMS subject classifications. 65F10, 65F25

1. Introduction.

The biconjugate gradient (BCG) algorithm [21] is the "natural" generalization of the classical conjugate gradient method [19] to nonsymmetric linear systems

$$Ax = b, \tag{1}$$

where $A \in R^{N \times N}$. It is an attractive method because of its simplicity and its good practical convergence properties. Unfortunately, one of its drawbacks is that it requires multiplications with the transpose matrix A^T. Methods have been developed which overcome this by computing residuals characterized by a product of the BCG residual polynomial with another polynomial of equal degree. Hence, we term the class of these algorithms *product methods.*

The conjugate gradients squared (CGS) algorithm (Sonneveld [26]) is a product method whose residuals can be written $r_n^{CGS} = \phi_n^2(A)r_0$, where $\phi_n(A)r_0$ is the residual from the standard BCG method. However, as discussed in [29], the convergence behavior of CGS can sometimes be irregular (i.e., $\|r_i^{CGS}\|$ can vary wildly with i). This is due to the fact that if $\phi_n(A)$ is viewed as a reduction operator applied twice to r_0, since $\phi_n(A)$ is quite dependent on the initial residual r_0, it may be possible that it is not a reduction for any other vector, not even for $\phi_n(A)r_0$ itself.

The Bi-CGSTAB algorithm [28] due to Van der Vorst attempts to stabilize this by multiplying the BCG polynomial $\phi_n(A)$, instead, by another polynomial of equal degree,

$$\tau_n(A) = (I - \omega_1 A)(I - \omega_2 A) \cdots (I - \omega_n A), \tag{2}$$

*Received by the editors June 6, 1994; accepted for publication (in revised form) June 27, 1995. This research was partially supported by ONR grant N00014-92-J-1890, NSF grant ASC92-01266, and ARO grant DAAL03-91-G-150. This paper was presented at the Colorado Conference on Iterative Methods, Breckenridge, CO, 1994.

†Department of Mathematics, University of California at Los Angeles, Los Angeles, CA 90024 (chan@math.ucla.edu, szeto@math.ucla.edu).

where the ω_i's are chosen to locally minimize the residual by a steepest descent method. Thus, by computing residuals $r_n^{Bi\text{-}CGSTAB} = \tau_n(A)\phi_n(A)r_0$, we obtain a more smoothly converging algorithm.

Unfortunately, problems arise in this method if we encounter a matrix A having complex eigenvalues with large imaginary parts. Since τ_n has only real zeros, it is difficult to accurately handle such eigenvalues. Thus, the Bi-CGSTAB2 algorithm (Gutknecht [18]) was developed to overcome this by performing two steps of Bi-CGSTAB at a time to allow for pairs of complex conjugate zeros but doing the local minimization at the nth step over the two degrees of freedom in ω_n and ω_{n-1}. Bi-CGSTAB2, then, is also a product method.

In the case that A is (nearly) skew symmetric, however, Bi-CGSTAB2 will suffer (near) breakdown due to the τ_{n-1} polynomial. The Bi-CGSTAB(2) method (a case of the Bi-CGSTAB(l) algorithm by Sleijpen and Fokkema [25] when $l = 2$) cures this by not involving τ_{n-1} in the intermediate step.

Since all of the product methods mentioned above involve the BCG residual polynomial ϕ_n, they not only inherit the good properties of BCG, but they also take on some of the problems of BCG. Specifically, it is well known that BCG suffers numerical breakdowns (attempts to divide by zero). There are two different possibilities of breakdown in the algorithm and many methods have been designed to cure them by "looking ahead" to avoid computing iterates where a breakdown can be predicted. The spectrum of these methods ranges from simple modifications of BCG to handle only one of the breakdowns to more complex algorithms which provide total breakdown protection using a variable look-ahead step. (See, e.g., [3, 5, 7, 8, 14, 16, 17, 20, 24].)

In this paper, we consider the *composite step* technique (Bank and Chan [2, 3]) which cures one of the breakdowns (assuming the other one does not occur) by simply looking ahead only one step when a (near) breakdown occurs. This technique is attractive because it does not require user-specified tolerance parameters or variable step sizes, and it is accomplished with only a minimal modification of the standard BCG algorithm. Moreover, the composite step technique can easily be extended to product methods. For example, the composite step CGS (CSCGS) algorithm was developed by Chan and Szeto in [9], and in this paper we show how composite step can be applied to Bi-CGSTAB.

In [3], Bank and Chan also prove a "best approximation" result which establishes a bound on the error of BCG. Here, we extend this result to prove convergence results for CGS, Bi-CGSTAB, and their composite step variants since these product methods all involve the BCG polynomial ϕ_n.

Section 2 describes in detail the composite step idea as originally presented for BCG and in §3 this is extended to Bi-CGSTAB yielding the method CS-CGSTAB. We emphasize in this section the strategy used to decide when to take a look-ahead composite step. We note that this stepping strategy not only skips exact breakdowns but is also designed to yield smoother residuals and does not require any user-specified tolerance parameters. A variant of this method, CS-CGSTAB2, which is a way of combining composite step with an idea from Bi-CGSTAB2, is presented in §4. The purpose of this variant is to cure additional breakdowns that may occur due to instability in the Bi-CGSTAB steepest descent polynomial in a manner similar to Bi-CGSTAB(l) with $l = 2$. The advantage is that our method cures pivot breakdowns as well. In addition, the step sizes in our new methods can vary as opposed to steps of fixed size in Bi-CGSTAB2 and Bi-CGSTAB(l). In §5, we show some numerical examples which compare these methods and show the advantages over their noncomposite step counterparts. Finally, §6 details the proofs for convergence of CGS, Bi-CGSTAB, and their composite step variants.

2. The CS idea: CSBCG. It is well known that the BCG method [19] is closely related to the nonsymmetric Lanczos process for computing the basis for the Krylov subspaces $K_n(r_0)$ and $K_n^*(\tilde{r}_0)$, defined as follows:

$$K_n(r_0) \equiv \text{span}\{r_0, Ar_0, \ldots, A^{n-1}r_0\}, \tag{3}$$

$$K_n^*(\tilde{r}_0) \equiv \text{span}\{\tilde{r}_0, A^T\tilde{r}_0, \ldots, (A^T)^{n-1}\tilde{r}_0\}. \tag{4}$$

The BCG iterates are defined by a Galerkin method on the associated Krylov subspaces. Given initial guesses of x_0 and $\tilde{x}_0$ to the solutions of (1) and an auxiliary system, $A^T\tilde{x} = \tilde{b}$, BCG produces iterates $x_n = x_0 + y_n$ and $\tilde{x}_n = \tilde{x}_0 + \tilde{y}_n$, with corresponding residuals of the form $r_n = b - Ax_n$ and $\tilde{r}_n = \tilde{b} - A^T\tilde{x}_n$, where $y_n \in K_n(r_0)$ and $\tilde{y}_n \in K_n^*(\tilde{r}_0)$, and such that the following Galerkin conditions are satisfied:

$$r_n \perp K_n^*(\tilde{r}_0), \qquad \tilde{r}_n \perp K_n(r_0). \tag{5}$$

If we define $\bar{K}_n$, $\bar{K}_n^*$ to be matrices whose columns are as given in (3) and (4) and span the Krylov spaces $K_n(r_0)$ and $K_n^*(\tilde{r}_0)$, respectively, condition (5) implies that the BCG iterates $x_n = x_0 + \bar{K}_n v_n$ are defined by the solution to the linear system $(\bar{K}_n^*)^T A\bar{K}_n v_n = (\bar{K}_n^*)^T r_0$. We note that the iterate x_n exists whenever the Hankel moment matrix

$$H_n^{(1)} = (\bar{K}_n^*)^T A\bar{K}_n = \begin{pmatrix} \mu_1 & \mu_2 & \cdots & \mu_n \\ \mu_2 & \mu_3 & \cdots & \mu_{n+1} \\ \vdots & \vdots & & \vdots \\ \mu_n & \mu_{n+1} & \cdots & \mu_{2n-1} \end{pmatrix},$$

where $\mu_i \equiv \tilde{r}_0^T A^i r_0$, is nonsingular.

In the standard BCG algorithm [21, 13], there are two possible kinds of numerical breakdowns (attempts to divide by zero): (1) $\sigma_n = \tilde{p}_n^T Ap_n = 0$, where p_n and $\tilde{p}_n$ are the BCG search directions *(pivot breakdown)*, and (2) $\rho_n = \tilde{r}_n^T r_n = 0$, but $r_n \neq 0$ *(Lanczos breakdown)*.[1] Although such exact breakdowns are very rare in practice, near breakdowns can cause severe numerical instability.

We term the first kind of breakdown a *pivot breakdown* because it is due to the nonexistence of the residual polynomial implicitly caused by encountering a zero pivot in the factorization of the tridiagonal matrix generated in the underlying Lanczos process. In terms of formally orthogonal polynomials [5], the BCG polynomial ϕ_n (defined from $r_n = \phi_n(A)r_0$) exists and is unique if and only if the $H_n^{(1)}$ is nonsingular. In other words, a pivot breakdown will occur at the nth iteration of the BCG algorithm if $\det(H_n^{(1)}) = 0$.

The second source of breakdown, *Lanczos breakdown*, is directly related to the breakdown of the underlying Lanczos process and is tied to the singularity of another Hankel moment matrix $H_n^{(0)}$ [17] defined by

$$H_n^{(0)} \equiv (\bar{K}_n^*)^T \bar{K}_n = \begin{pmatrix} \mu_0 & \mu_1 & \cdots & \mu_{n-1} \\ \mu_1 & \mu_2 & \cdots & \mu_n \\ \vdots & \vdots & & \vdots \\ \mu_{n-1} & \mu_n & \cdots & \mu_{2n-2} \end{pmatrix}.$$

[1] In other literature, what we term the *pivot* and *Lanczos* breakdowns are also known as *true* and *ghost* breakdowns [6], *Galerkin* and *serious Lanczos* breakdowns [14], and *hard* and *soft* breakdowns [20].

There are many methods designed to handle pivot breakdowns (see, e.g., [3, 14, 17]), as well as methods which cure both types of breakdown (see, e.g., [5, 7, 8, 14, 15, 16, 20, 24]). Although the step size needed to overcome an exact Lanczos breakdown can be computed in principle, these methods can unfortunately be quite complicated for handling near breakdowns since the sizes of the look-ahead steps are variable (indeed, the breakdowns can be *incurable*).

The CSBCG algorithm (Bank and Chan [2, 3]) is an alternative which cures only the pivot breakdown (assuming no Lanczos breakdowns) by "looking ahead" in $H_n^{(1)}$ and not computing x_n where it is not defined. Looking ahead, in general, means that we build $H_i^{(1)}$ until it is no longer singular and we have an iterate $x_{n+m}, m \geq 1$. In CSBCG, this is done with a simple modification of BCG which needs only a maximum look-ahead step size of $m = 2$ to eliminate the (near) breakdown and to smooth the sometimes erratic convergence of BCG. It was shown in [2, Lem. 4.3] that only two steps are needed if we assume no Lanczos breakdown, but this can also be seen in the relationship between the two Hankel matrices defined above: assuming that $\det(H_n^{(0)}) \neq 0$ for all n, then no two consecutive principal submatrices of $H_n^{(1)}$ can be singular. (The structure of these Hankel determinants was studied in detail by Draux [10].) Thus, instead of a more complicated (but less prone to breakdown) version, CSBCG cures only one kind of breakdown, but does so with a minimal modification to the usual implementation of BCG in the hope that its empirically observed stability will be inherited [27].

Next, we shall briefly review some of the details of CSBCG. Suppose that in running the BCG algorithm we encounter a situation where $\sigma_n = 0$ at step n and, therefore, the values $x_{n+1}, \tilde{x}_{n+1}, r_{n+1}, \tilde{r}_{n+1}$ cannot be defined. The composite step approach is to overcome this problem by skipping the $n+1$ update and computing the quantities in step $n+2$ by using scaled versions of r_{n+1} and $\tilde{r}_{n+1}$, which do not require divisions by σ_n. More specifically, we define the auxiliary vector:

$$z_{n+1} \equiv \sigma_n r_{n+1} = \sigma_n r_n - \rho_n A p_n \in K_{n+2}(r_0); \tag{6}$$

z_{n+1} and its counterpart $\tilde{z}_{n+1} \equiv \sigma_n \tilde{r}_{n+1}$ exist even if $\sigma_n = 0$ and, thus, can be used in defining x_{n+2}:

$$x_{n+2} = x_n + [p_n, z_{n+1}] f_n,$$

with corresponding residual and search direction

$$r_{n+2} = r_n - A[p_n, z_{n+1}] f_n, \tag{7}$$

$$p_{n+2} = r_{n+2} + [p_n, z_{n+1}] g_n, \tag{8}$$

where $f_n, g_n \in R^2$, and similarly for $\tilde{x}_{n+2}, \tilde{r}_{n+2}, \tilde{p}_{n+2}, \tilde{f}_n$, and $\tilde{g}_n$.

To solve for the unknowns $f_n = (f_n^{(1)}, f_n^{(2)})^T$ and $g_n = (g_n^{(1)}, g_n^{(2)})^T$, we impose the Galerkin condition and conjugacy condition of BCG which result in two 2×2 linear systems:

$$\begin{bmatrix} \tilde{p}_n^T A p_n & \tilde{p}_n^T A z_{n+1} \\ \tilde{z}_{n+1}^T A p_n & \tilde{z}_{n+1}^T A z_{n+1} \end{bmatrix} \begin{bmatrix} f_n^{(1)} \\ f_n^{(2)} \end{bmatrix} = \begin{bmatrix} \tilde{p}_n^T r_n \\ \tilde{z}_{n+1}^T r_n \end{bmatrix}, \tag{9}$$

$$\begin{bmatrix} \tilde{p}_n^T A p_n & \tilde{p}_n^T A z_{n+1} \\ \tilde{z}_{n+1}^T A p_n & \tilde{z}_{n+1}^T A z_{n+1} \end{bmatrix} \begin{bmatrix} g_n^{(1)} \\ g_n^{(2)} \end{bmatrix} = - \begin{bmatrix} \tilde{p}_n^T A r_{n+2} \\ \tilde{z}_{n+1}^T A r_{n+2} \end{bmatrix}. \tag{10}$$

These systems can be solved simply, as shown in [3], making it possible to compute x_{n+2}, $\tilde{x}_{n+2}, r_{n+2}, \tilde{r}_{n+2}$ and, thus, advance from step n to step $n+2$. The CSBCG algorithm, then, is simply the combination of the 1×1 and 2×2 steps.

3. Composite step Bi-CGSTAB. We now apply the composite step idea to handle the pivot breakdowns that occur in the Bi-CGSTAB method proposed by Van der Vorst [28]. Since the Bi-CGSTAB residual polynomials are formed from multiplying the BCG polynomial ϕ_n with another polynomial τ_n of the form (2), we can use the subset of well-defined ϕ_i's from CSBCG to multiply with τ_n instead. To do this, we first define

$$p_n^{BCG} = \phi_n(A)r_0, \quad z_{n+1}^{BCG} = \xi_{n+1}(A)r_0.$$

The CS-CGSTAB polynomials, then, take on the form

$$r_n^{CS\text{-}CGSTAB} = \tau_n(A)\phi_n(A)r_0, \quad p_n^{CS\text{-}CGSTAB} = \tau_n(A)\psi_n(A)r_0.$$

In the case of a 1 × 1 step, we simply use the Bi-CGSTAB update. For the 2 × 2 composite step, we will need to evaluate

$$\begin{aligned} r_{n+2}^{CS\text{-}CGSTAB} &= \tau_{n+2}\phi_{n+2}r_0 \\ &= (I - \omega_{n+2}A)(I - \omega_{n+1}A)\tau_n\phi_{n+2}r_0 \end{aligned} \tag{11}$$

and

$$\begin{aligned} p_{n+2}^{CS\text{-}CGSTAB} &= \tau_{n+2}\psi_{n+2}r_0 \\ &= (I - \omega_{n+2}A)(I - \omega_{n+1}A)\tau_n\psi_{n+2}r_0 \end{aligned} \tag{12}$$

using the quantities obtained from the nth step: $\tau_n\phi_n$ and $\tau_n\psi_n$. We first show how to compute the polynomials $\tau_n\phi_{n+2}$ and $\tau_n\psi_{n+2}$ appearing in (11)–(12). We use the CSBCG relationships from (6)–(8):

$$\xi_{n+1}(\vartheta) = \sigma_n\phi_n - \rho_n\vartheta\psi_n, \tag{13}$$

$$\phi_{n+2}(\vartheta) = \phi_n - \vartheta\psi_n f_n^{(1)} - \vartheta\xi_{n+1}f_n^{(2)}, \tag{14}$$

$$\psi_{n+2}(\vartheta) = \phi_{n+2} + \psi_n g_n^{(1)} + \xi_{n+1}g_n^{(2)} \tag{15}$$

and multiply by the τ_n polynomial to evaluate the needed quantities:

$$\tau_n(\vartheta)\xi_{n+1}(\vartheta) = \sigma_n\tau_n\phi_n - \rho_n\vartheta\tau_n\psi_n, \tag{16}$$

$$\begin{aligned} \tau_n(\vartheta)\phi_{n+2}(\vartheta) &= \tau_n(\phi_n - \vartheta\psi_n f_n^{(1)} - \vartheta\xi_{n+1}f_n^{(2)}) \\ &= \tau_n\phi_n - \vartheta\tau_n\psi_n f_n^{(1)} - \vartheta\tau_n\xi_{n+1}f_n^{(2)}, \end{aligned} \tag{17}$$

$$\begin{aligned} \tau_n(\vartheta)\psi_{n+2}(\vartheta) &= \tau_n(\phi_{n+2} + \psi_n g_n^{(1)} + \xi_{n+1}g_n^{(2)}) \\ &= \tau_n\phi_{n+2} + \tau_n\psi_n g_n^{(1)} + \tau_n\xi_{n+1}g_n^{(2)}. \end{aligned} \tag{18}$$

We now show how to compute the unknowns f_n and g_n in (17) and (18). The CSBCG residual r_{n+2} in (7) and search direction p_{n+2} in (8) are, respectively, orthogonal and A-conjugate to $K_{n+1}^*(\tilde{r}_0)$ [2]. By imposing orthogonality and A-conjugacy conditions on two specific vectors $\tau_n(A^T)\tilde{r}_0 \in K_{n+1}^*(\tilde{r}_0)$ and $A^T\tau_n(A^T)\tilde{r}_0 \in K_{n+1}^*(\tilde{r}_0)$, we obtain two linear systems which give f_n and g_n.

Specifically, by writing (7) in polynomial form (14) and taking an inner product with $\tau_n(A^T)\tilde{r}_0$, we obtain the relation

$$\begin{aligned} 0 &= (\tau_n(A^T)\tilde{r}_0, \phi_{n+2}(A)r_0) \\ &= (\tilde{r}_0, \tau_n(A)\phi_{n+2}(A)r_0) \\ &= (\tilde{r}_0, [\tau_n\phi_n - A\tau_n\psi_n f_n^{(1)} - A\tau_n\xi_{n+1}f_n^{(2)}](A)r_0). \end{aligned}$$

Similarly, for the A-conjugacy of the search direction (8),

$$\begin{aligned} 0 &= (A^T\tau_n(A^T)\tilde{r}_0, \psi_{n+2}r_0) \\ &= (\tilde{r}_0, A\tau_n(A)\psi_{n+2}(A)r_0) \\ &= (\tilde{r}_0, [A\tau_n\phi_{n+2} + A\tau_n\psi_n g_n^{(1)} + A\tau_n\xi_{n+1}g_n^{(2)}](A)r_0). \end{aligned}$$

We derive two similar relations by imposing the orthogonality and A-conjugacy conditions on $A^T\tau_n(A^T)\tilde{r}_0$. Combining the four relations, we obtain the following two 2×2 linear systems:

$$\begin{bmatrix} (\tilde{r}_0, \vartheta\tau_n\psi_n r_0) & (\tilde{r}_0, \vartheta\tau_n\xi_{n+1}r_0) \\ (\tilde{r}_0, \vartheta^2\tau_n\psi_n r_0) & (\tilde{r}_0, \vartheta^2\tau_n\xi_{n+1}r_0) \end{bmatrix} \begin{bmatrix} f_n^{(1)} \\ f_n^{(2)} \end{bmatrix} = \begin{bmatrix} (\tilde{r}_0, \tau_n\phi_n r_0) \\ (\tilde{r}_0, \vartheta\tau_n\phi_n r_0) \end{bmatrix}, \tag{19}$$

$$\begin{bmatrix} (\tilde{r}_0, \vartheta\tau_n\psi_n r_0) & (\tilde{r}_0, \vartheta\tau_n\xi_{n+1}r_0) \\ (\tilde{r}_0, \vartheta^2\tau_n\psi_n r_0) & (\tilde{r}_0, \vartheta^2\tau_n\xi_{n+1}r_0) \end{bmatrix} \begin{bmatrix} g_n^{(1)} \\ g_n^{(2)} \end{bmatrix} = \begin{bmatrix} (\tilde{r}_0, \vartheta\tau_n\phi_{n+2}) \\ (\tilde{r}_0, \vartheta^2\tau_n\phi_{n+2}) \end{bmatrix}. \tag{20}$$

These are easily solved since all of the entries in the 2×2 matrix and the right-hand sides can be obtained from (16), (17), and quantities from the nth step. Thus, we can update (17) and (18). Note that the linear systems (19) and (20) can be shown to be nonsingular in the case where $\sigma_n = 0$ and the underlying Lanczos process does not break down. (See the Appendix.) This verifies that a 2×2 step is always sufficient for skipping over the pivot breakdown.

The next step in evaluating (11) and (12) is to choose ω_{n+1} and ω_{n+2} to satisfy some local minimization property. In the case of a 1×1 step, we imitate the Bi-CGSTAB update steps. Specifically, ω_{n+1} is chosen to minimize the norm of $r_{n+1} = \tau_{n+1}\phi_{n+1}r_0 = (I - \omega_{n+1}A)\tau_n\phi_{n+1}r_0$. For the 2×2 step, we employ the same steepest descent rule to compute ω_{n+1} and ω_{n+2} by minimizing $\|r_{n+1}\|$ and $\|r_{n+2}\|$. Note that r_{n+1} is not available in a 2×2 step, but we can use a scaled version for minimization. To do this, let $u_{n+1} = \tau_n\xi_{n+1}r_0 \equiv \frac{1}{\sigma_n}\tau_n\phi_{n+1}r_0$. Then we can write

$$r_{n+1} = \tau_{n+1}\phi_{n+1}r_0 = (I - \omega_{n+1}A)\tau_n\phi_{n+1}r_0 = (I - \omega_{n+1}A)\frac{1}{\sigma_n}\, u_{n+1}. \tag{21}$$

The vector u_{n+1} is already computed in the CS-CGSTAB algorithm (see relation (16)) and, thus, we minimize

$$\|r_{n+1}^2\| = \frac{1}{\sigma_n^2}\left((I - \omega_{n+1}A)u_{n+1}\right)^T\left((I - \omega_{n+1}A)u_{n+1}\right)$$

by choosing $\omega_{n+1} = (Au_{n+1}, u_{n+1})/(Au_{n+1}, Au_{n+1})$, an orthogonal projection of u_{n+1} onto Au_{n+1}.

Similarly, let

$$u_{n+2} \equiv \tau_{n+1}\phi_{n+2}r_0 = (I - \omega_{n+1}A)\tau_n\phi_{n+2}r_0 \tag{22}$$

which can be computed because $\tau_n\phi_{n+2}$ is available from relation (17). Then write

$$r_{n+2} = \tau_{n+2}\phi_{n+2}r_0 = (I - \omega_{n+2}A)\tau_{n+1}\phi_{n+2}r_0 = (I - \omega_{n+2}A)u_{n+2} \tag{23}$$

and minimize

$$\|r_{n+2}^2\| = \left((I - \omega_{n+2}A)u_{n+2}\right)^T\left((I - \omega_{n+2}A)u_{n+2}\right)$$

by choosing

$$\omega_{n+2} = (Au_{n+2}, u_{n+2})/(Au_{n+2}, Au_{n+2}). \tag{24}$$

Finally, in running CS-CGSTAB, we must be able to recover the BCG constants ρ_n^{BCG} and σ_n^{BCG} in order to update the BCG polynomial part of the residual. In [28], Van der Vorst defined the Bi-CGSTAB constant

$$\rho_n^{Bi\text{-}CGSTAB} = (\tilde{r}_0, r_n^{Bi\text{-}CGSTAB})$$

and showed its relation to

$$\rho_n^{BCG} = (\tilde{r}_n^{BCG}, r_n^{BCG}).$$

Using the property that, by construction, $\phi_n(A)r_0$ is orthogonal with respect to all vectors $\chi_{n-1}(A^T)\tilde{r}_0$, where χ_{n-1} is an arbitrary polynomial of degree at most $n-1$, one gets

$$\begin{aligned}
\rho_n^{BCG} &= (\phi_n(A^T)\tilde{r}_0, \phi_n(A)r_0)\\
&= \alpha_0 \ldots \alpha_{n-1}((-A^T)^n\tilde{r}_0, \phi_n(A)r_0),\\
\rho_n^{Bi\text{-}CGSTAB} &= (\tilde{r}_0, \tau_n(A)\phi_n(A)r_0)\\
&= (\tau_n(A^T)\tilde{r}_0, \phi_n(A)r_0)\\
&= \omega_1 \ldots \omega_n((-A^T)^n\tilde{r}_0, \phi_n(A)r_0).
\end{aligned}$$

The relationship between them follows:

$$\rho_{n+1}^{BCG} = \frac{\alpha_0 \ldots \alpha_n}{\omega_1 \ldots \omega_{n+1}} \rho_{n+1}^{Bi\text{-}CGSTAB}, \quad \text{where } \alpha_n = \frac{\rho_n^{BCG}}{\sigma_n^{BCG}}.$$

If we let $\mu_n = (\alpha_0 \ldots \alpha_{n-1})/(\omega_1 \ldots \omega_n)$, this reduces to the update formula

$$\rho_{n+1}^{BCG} = \mu_{n+1}\rho_{n+1}^{Bi\text{-}CGSTAB} = \mu_n \left(\frac{\rho_n^{BCG}}{\sigma_n^{BCG}\omega_{n+1}} \right) \rho_{n+1}^{Bi\text{-}CGSTAB}.$$

In a 2 × 2 step, we determine ρ_{n+2}^{CSBCG} similarly. First we use relation (14):

$$\begin{aligned}
\rho_{n+2}^{CSBCG} &= (\phi_{n+2}(A^T)\tilde{r}_0, \phi_{n+2}(A)r_0)\\
&= ((\phi_n(A^T) - \alpha_n A^T \psi_n(A^T) - \alpha_{n+1}A^T\xi_{n+1}(A^T))\tilde{r}_0, \phi_{n+2}(A)r_0).
\end{aligned} \tag{25}$$

Then the fact that $\phi_{n+2}r_0$ is orthogonal to all vectors $\chi_{n+1}(A^T)\tilde{r}_0$ implies that the inner product (25) picks out the coefficient of the highest-order term of the $n+2$ degree polynomial that ϕ_{n+2} is being orthogonalized against. In this case, the coefficient of the highest-order term for the polynomial $\xi_{n+1}(A^T) = \sigma_n^{BCG}\phi_{n+1}(A^T)$ is $\sigma_n^{BCG}\alpha_0 \ldots \alpha_n$, so (25) reduces to

$$\rho_{n+2}^{CSBCG} = -\alpha_0 \ldots \alpha_{n+1}\sigma_n^{BCG}((-A^T)^{n+2}\tilde{r}_0, \phi_{n+2}(A)r_0)$$

which leads to the update formula

$$\rho_{n+2}^{CSBCG} = \mu_{n+2}\rho_{n+2}^{CS\text{-}CGSTAB} = -\mu_n \left(\frac{\rho_n^{CSBCG}\alpha_{n+1}}{\omega_{n+1}\omega_{n+2}} \right) \rho_{n+2}^{CS\text{-}CGSTAB}. \tag{26}$$

The update for $\sigma_n^{BCG} \equiv (\psi_n(A^T)\tilde{r}_0, A\psi_n(A)r_0)$ can be calculated similarly. We define $\sigma_n^{CS\text{-}CGSTAB} \equiv (\tilde{r}_0, A\tau_n(A)\psi_n(A)r_0) = (\tilde{r}_0, Ap_n^{CS\text{-}CGSTAB})$ and obtain the analogous update formula

$$\sigma_{n+2}^{CSBCG} = \mu_{n+2}\sigma_{n+2}^{CS\text{-}CGSTAB}. \tag{27}$$

TABLE 1
Notation for CS-CGSTAB.

Vector	Polynomial	Where/how derived	Where used
r_n	$\tau_n\phi_n r_0$	(11)	(16) (17) (19)
p_n	$\tau_n\psi_n r_0$	(12)	(18)
u_{n+1}	$\tau_n\xi_{n+1} r_0$	(16)	(17) (18) (21)
s_{n+2}	$\tau_n\phi_{n+2} r_0$	(17)	(11) (18) (22)
u_{n+2}	$\tau_{n+1}\phi_{n+2} r_0$	(22)	(23)
e_n	$A\tau_n\phi_n r_0$	Ar_n	(19)
q_n	$A\tau_n\psi_n r_0$	Ap_n	(16) (17) (19) (20)
y_{n+1}	$A\tau_n\xi_{n+1} r_0$	Au_{n+1}	(19) (20) (21)
t_{n+2}	$A\tau_n\phi_{n+2} r_0$	As_{n+2}	(20) (22)
y_{n+2}	$A\tau_{n+1}\phi_{n+2} r_0$	Au_{n+2}	(23)
c_n	$A^2\tau_n\psi_n r_0$	Aq_n	(19) (20)
d_{n+1}	$A^2\tau_n\xi_{n+1} r_0$	Ay_{n+1}	(19) (20)
v_{n+2}	$A^2\tau_n\phi_{n+2} r_0$	At_{n+2}	(20)
α_n	$f_n^{(1)}$	(19)	(17)
α_{n+1}	$f_n^{(2)}$	(19)	(17)
β_n	$g_n^{(1)}$	(20)	(18)
β_{n_1}	$g_n^{(2)}$	(20)	(18)

3.1. Implementation details. In Table 1, we summarize the notation we will be using in our implementation of the CS-CGSTAB algorithm. We list the vector used in the algorithm, its corresponding polynomial form, and the equations in which it is derived and used.

At every step, we must anticipate a 2 × 2 step even if we decide to take a 1 × 1 step. (The stepping strategy will be discussed in §3.2.) Recall that in one step of Bi-CGSTAB, only two matrix–vector multiplications are performed: $q_n = Ap_n$ and $y_{n+1} = Au_{n+1}$. From the third column of Table 1, we see that eight matrix–vector products are required to do a 2 × 2 step which appears to be four more than two steps of Bi-CGSTAB.

However, the total eight products can be reduced to four multiplications by precomputing certain values and absorbing them into vector updates rather than explicitly multiplying by A. Specifically, by precomputing $c_n \equiv Aq_n$, the multiplication $y_{n+1} \equiv Au_{n+1}$ can be written

$$\begin{aligned} y_{n+1} &= \sigma_n e_n - \rho_n A q_n \\ &= \sigma_n e_n - \rho_n c_n. \end{aligned}$$

Similarly, if we have $d_{n+1} \equiv Ay_{n+1}$, then the product $e_{n+1} \equiv Ar_{n+1}$ can also be evaluated without having to multiply by A:

$$\begin{aligned} e_{n+1} &= (y_{n+1} - \omega_{n+1} A y_{n+1})/\sigma_n \\ &= (y_{n+1} - \omega_{n+1} d_{n+1})/\sigma_n. \end{aligned}$$

Furthermore, $q_{n+1} \equiv Ap_{n+1}$ can also be updated:

$$\begin{aligned} q_{n+1} &= Ar_{n+1} + \beta_{n+1}(Ap_n - \omega_{n+1} A q_n) \\ &= e_{n+1} + \beta_{n+1}(q_n - \omega_{n+1} c_n). \end{aligned}$$

Thus, the 1 × 1 step can still be performed with only two (pre)multiplications with A: Aq_n and Ay_{n+1}. Moreover, by precomputing $v_{n+2} \equiv At_{n+2}$ and $q_{n+2} \equiv Ap_{n+2}$, we can update the remaining values in the 2 × 2 step in a similar fashion.

However, using this precomputing strategy to update

$$e_{n+2} \equiv Ar_{n+2} = A(I - \omega_{n+1}A)(I - \omega_{n+2}A)s_{n+2}$$

implies that we need $A^3 s_{n+2}$ which we do not have. Thus, the 2×2 step requires an additional matrix–vector multiplication: $w_{n+2} \equiv A^3 s_{n+2} = A(v_{n+2})$.

Hence, as in CSCGS, there are five matrix–vector multiplications for a 2×2 step, whereas in two steps of Bi-CGSTAB, only four are needed. This is the price we must pay for composite step. If we do not need to take many 2×2 steps in practice, this price will not be too costly.

3.2. CS-CGSTAB stepping strategy. As far as deciding when to actually take a 2×2 step, we follow the principles in [2, 3, 9]. As with these methods, CS-CGSTAB employs a practical stepping strategy that will skip over exact breakdowns using the criterion

$$\|r_{n+1}\| > \max\{\|r_n\|, \|r_{n+2}\|\}. \tag{28}$$

If this condition were met (e.g., at a near breakdown) and we performed two 1×1 steps, it would result in a "peak" in the residual convergence. By taking a 2×2 step, we skip over this and obtain a smoother, more stable method. In order to avoid unnecessary computation of $\|r_{n+2}\|$, we express condition (28) in the following algorithm.

```
If ($\|r_{n+1}\| < \|r_n\|$) then          ⟵ Condition (28a)
   choose $1 \times 1$ step
else
   if ($\|r_{n+1}\| < \|r_{n+2}\|$) then    ⟵ Condition (28b)
      choose $1 \times 1$ step
   else
      choose $2 \times 2$ step
   end
end
```

In order to avoid repeating work and to do this in a stable way, Condition (28a) can be written

$$\|(I - \omega_{n+1} A) u_{n+1}\| < |\sigma_n| \|r_n\|.$$

For Condition (28b), we first rescale the r_{n+2} update in order to estimate $\|r_{n+2}\|$ stably by letting

$$\nu_{n+2} \equiv \delta_n r_{n+2} = \delta_n (I - \omega_{n+1} A)(I - \omega_{n+2} A) s_{n+2},$$

where δ_n is the determinant of the 2×2 matrix in (19). Evaluating ν_{n+2} exactly would involve the quantity $A^2 s_{n+2}$ which would require an additional matrix–vector multiplication if we decide to take a 1×1 step. Hence, for practical purposes, we use an upper bound approximation to estimate $\|\nu_{n+2}\|$. Note that although we do not have $A^2 s_{n+2}$, the quantity $A s_{n+2}$ can be made available even if we take a 1×1 step, and it can be done without an additional matrix–vector product using the precomputed values e_n, c_n, and d_{n+1}:

$$\begin{aligned} t_{n+2} &\equiv A s_{n+2} \\ &= A(\delta_n r_n - \alpha_n q_n - \alpha_{n+1} y_{n+1}) \\ &= \delta_n e_n - \alpha_n c_n - \alpha_{n+1} d_{n+1}. \end{aligned} \tag{29}$$

Thus, we would like to estimate $\|\nu_{n+2}\| = \|\delta_n (I - \omega_{n+1} A)(I - \omega_{n+2} A) s_{n+2}\|$ using (29) and without any further matrix–vector multiplications. Our strategy is to set $\omega_{n+2} = 0$ and minimize $\|\delta_n (I - \omega_{n+1} A) s_{n+2}\|$, thereby eliminating the need to compute $A^2 s_{n+2}$. Doing this gives an upper bound estimate to $\|\nu_{n+2}\|$ which can be shown by defining

$$h(\omega_{n+1}, \omega_{n+2}) \equiv \|\delta_n (I - \omega_{n+1} A)(I - \omega_{n+2} A) s_{n+2}\|$$

and establishing the following inequality:

$$\min_{\omega_{n+1},\,\omega_{n+2}} h(\omega_{n+1}, \omega_{n+2}) \le \min_{\omega_{n+1}} h(\omega_{n+1}, 0). \tag{30}$$

The minimization problem

$$\min_{\omega_{n+1}} h(\omega_{n+1}, 0) = \min_{\omega_{n+1}} \|(I - \omega_{n+1} A) s_{n+2}\| = \min_{\omega_{n+1}} \|s_{n+2} - \omega_{n+1} t_{n+2}\|$$

is solved with $\tilde{\omega}_{n+1} = (t_{n+2}, s_{n+2})/(t_{n+2}, t_{n+2})$.

Hence, Condition (28b), $\|r_{n+1}\| < \|r_{n+2}\|$, is evaluated by the approximated condition

$$|\delta_n| \|(I - \omega_{n+1} A) u_{n+1}\| < |\sigma_n| \|\tilde{v}_{n+2}\|, \tag{31}$$

where $\|\tilde{v}_{n+2}\|$ is the estimated upper bound for $\|v_{n+2}/\delta_n\|$:

$$\|v_{n+2}/\delta_n\| < \|\tilde{v}_{n+2}\| \equiv \|s_{n+2} - \tilde{\omega}_{n+1} t_{n+2}\|. \tag{32}$$

If a 2 × 2 step is chosen, then the true v_{n+2} is evaluated. After v_{n+2} is computed, we add one final check to make sure the approximation was indeed valid. If not, we take a 1 × 1 step. In Table 2, we present the CS-CGSTAB algorithm. Note that if only 1 × 1 steps are taken, we have exactly the Bi-CGSTAB algorithm.

4. A variant of CS-CGSTAB. A problem in the CS-CGSTAB method is that additional breakdowns may occur if A is skew symmetric or near breakdowns if it has complex eigenvalues with large imaginary parts. Suppose we are at step n of the algorithm. In the case that A is skew symmetric, it can be easily checked that $\omega_{n+1} = \omega_{n+2} = 0$. The 2 × 2 step attempts to divide by the quantity $\gamma_2 = \omega_{n+1}\omega_{n+2}$ which causes a breakdown. For nearly skew symmetric A, γ_2 will be small, thus causing near breakdown and numerical instability.

This can be cured by modifying the local minimization at that particular 2 × 2 step. The idea is to require

$$\|r_{n+2}\| = \min_{\varphi \in \mathcal{P}_2} \|\varphi(A) \tau_n(A) \phi_{n+2}(A) r_0\|, \tag{33}$$

where $\mathcal{P}_2$ is the set of all polynomials of degree (at most) 2 and $\varphi(0) = 1$.

Performing this minimization over two degrees of freedom was first presented in the Bi-CGSTAB2 algorithm by Gutknecht [18]. The purpose was to cure problems that arise in the steepest descent part of Bi-CGSTAB due to eigenvalues of A in the complex spectrum that are not approximated well with (2). Hence, every other step of the Bi-CGSTAB2 method performs (33) in order to handle conjugate pairs of eigenvalues. (In the remaining steps of the Bi-CGSTAB2 algorithm, the usual Bi-CGSTAB update is taken.) The Bi-CGSTAB2 algorithm can be summarized:

$$\tau_n \phi_n \xrightarrow{\text{1-D min}} \tau_{n+1} \phi_{n+1} \xrightarrow{\text{2-D min}} \tau_{n+2} \phi_{n+2}.$$

Unfortunately, in the implementation presented in [18], the 2-D minimization steps of Bi-CGSTAB2 are computed based on the Bi-CGSTAB step immediately before them. For skew-symmetric A, this poses a similar breakdown problem to the one mentioned above because the Bi-CGSTAB step requires a division by ω_{n+1}, which will be zero in this case.

Note that Bi-CGSTAB2 is mathematically equivalent to the Bi-CGSTAB(l) algorithm due to Sleijpen and Fokkema [25] in the case where $l = 2$, but the implementation is different. Bi-CGSTAB(2) does not compute the intermediate Bi-CGSTAB residual $r_{n+1} = \tau_{n+1} \phi_{n+1} r_0$.

TABLE 2
Algorithm CS-CGSTAB.

```
  $\rho_0 = \tilde{r}_0^T r_0$;   $p_0 = r_0$;   $\phi_0 = \|r_0\|$;   $e_0 = q_0 = Ar_0$;   $\mu_0 = 1$
  $n \leftarrow 0$
While *method not converged yet* do:
  $\sigma_n = (\tilde{r}_0^T q_n)\mu_n$;   $c_n = Aq_n$                                         % Evaluate (27)
  $u_{n+1} = \sigma_n r_n - \rho_n q_n$;   $y_{n+1} = \sigma_n e_n - \rho_n c_n$;   $d_{n+1} = Ay_{n+1}$
  $\omega_{n+1} = (y_{n+1}, u_{n+1})/(y_{n+1}, y_{n+1})$
  $\hat{r}_{n+1} = u_{n+1} - \omega_{n+1} y_{n+1}$;   $\hat{e}_{n+1} = y_{n+1} - \omega_{n+1} d_{n+1}$;   $\psi_{n+1} = \|\hat{r}_{n+1}\|$
  % Decide whether to take a $1 \times 1$ step or a $2 \times 2$ step.
    If $\psi_{n+1} < |\sigma_n|\phi_n$, Then      % Condition (28a): $\|r_{n+1}\| < \|r_n\|$
      *one-step* = 1
    Else
      $a_{11} = \tilde{r}_0^T q_n$;   $a_{12} = \tilde{r}_0^T y_{n+1}$;   $a_{21} = \tilde{r}_0^T c_n$;   $a_{22} = \tilde{r}_0^T d_{n+1}$
      $\delta_n = a_{11}a_{22} - a_{12}a_{21}$;   $b_1 = \rho_n/\mu_n$;   $b_2 = \tilde{r}_0^T e_{n+1}$
      $\alpha_n = a_{22}b_1 - a_{12}b_2$;   $\alpha_{n+1} = -a_{21}b_1 + a_{11}b_2$                  % Solve (19)
      $s_{n+2} = \delta_n r_n - \alpha_n q_n - \alpha_{n+1} y_{n+1}$;   $t_{n+2} = \delta_n e_n - \alpha_n c_n - \alpha_{n+1} d_{n+1}$
      $\tilde{\omega}_{n+1} = (t_{n+2}, s_{n+2})/(t_{n+2}, t_{n+2})$;   $\tilde{\nu}_{n+2} = \|s_{n+2} - \tilde{\omega}_{n+1} t_{n+2}\|$   % Evaluate (32)
      If $|\delta_n|\psi_{n+1} < |\sigma_n|\tilde{\nu}_{n+2}$, Then      % Condition (28b): $\|r_{n+1}\| < \|r_{n+2}\|$
          *one-step* = 1
      Else                                                                  % True $\nu_{n+2}$
          $\upsilon_{n+2} = At_{n+2}$;   $w_{n+2} = A\upsilon_{n+2}$;   $z_{n+2} = t_{n+2} - \omega_{n+1}\upsilon_{n+2}$
          $\omega_{n+2} = (z_{n+2}, u_{n+2})/(z_{n+2}, z_{n+2})$;   $\gamma_1 = -(\omega_{n+1} + \omega_{n+2})$;   $\gamma_2 = \omega_{n+1}\omega_{n+2}$
          $\hat{r}_{n+2} = s_{n+2} + \gamma_1 t_{n+2} + \gamma_2 \upsilon_{n+2}$;   $\nu_{n+2} = \|\hat{r}_{n+2}\|$
          $\hat{e}_{n+2} = t_{n+2} + \gamma_1 \upsilon_{n+2} + \gamma_2 w_{n+2}$
          If $|\delta_n|\psi_{n+1} < |\sigma_n|\nu_{n+2}$, Then                         % Retest w/$\nu_{n+2}$
              *one-step* = 1
          Else
              *one-step* = 0
    End If;   End If;   End If
  % Compute next iterate.
    If *one-step*, Then        % *** Usual Bi-CGSTAB ***
      $r_{n+1} = \hat{r}_{n+1}/\sigma_n$;   $e_{n+1} = \hat{e}_{n+1}/\sigma_n$;   $\phi_{n+1} = \psi_{n+1}/\sigma_n$
      $x_{n+1} = x_n + (\rho_n p_n + \omega_{n+1} u_{n+1})/\sigma_n$
      $\mu_{n+1} = (\mu_n \rho_n)/(\sigma_n \omega_{n+1})$;   $\rho_{n+1} = (\tilde{r}_0^T r_{n+1})\mu_{n+1}$
      $\beta_{n+1} = \rho_{n+1}/\rho_n$
      $p_{n+1} = r_{n+1} + \beta_{n+1}(p_n - \omega_{n+1} q_n)$
      $q_{n+1} = e_{n+1} + \beta_{n+1}(q_n - \omega_{n+1} c_n)$
      $n \leftarrow n + 1$
    Else        % *** $2 \times 2$ step CS-CGSTAB ***
      $r_{n+2} = \hat{r}_{n+2}/\delta_n$;   $e_{n+2} = \hat{e}_{n+2}/\delta_n$;   $\phi_{n+2} = \nu_{n+2}/\delta_n$
      $x_{n+2} = x_n + (\alpha_n p_n + \alpha_{n+1} u_{n+1} - \gamma_1 s_{n+2} - \gamma_2 t_{n+2})/\delta_n$
      $\mu_{n+2} = -\mu_n(\alpha_{n+1}\rho_n/\delta_n\gamma_2)$;   $\rho_{n+2} = (\tilde{r}_0^T r_{n+2})\mu_{n+2}$       % Evaluate (26)
      $b_1 = \tilde{r}_0^T t_{n+2}$;   $b_2 = \tilde{r}_0^T \upsilon_{n+2}$
      $\beta_n = (a_{22}b_1 - a_{12}b_2)/\delta_n$;   $\beta_{n+1} = (-a_{21}b_1 + a_{11}b_2)/\delta_n$      % Solve (20)
      $p_{n+2} = r_{n+2} - \beta_n(p_n + \gamma_1 q_n + \gamma_2 c_n) - \beta_{n+1}(u_{n+1} + \gamma_1 y_{n+1} + \gamma_2 d_{n+1})$
      $q_{n+2} = Ap_{n+2}$
      $n \leftarrow n + 2$
    End If
End While
```

Instead, it uses an intermediate basis vector $A\tau_n\phi_{n+1}r_0$ to generate the auxiliary residual $\hat{r}_{n+2} = \tau_n\phi_{n+2}r_0$. This algorithm can be summarized:

$$\tau_n\phi_n \xrightarrow{A\tau_n\phi_{n+1}} \tau_n\phi_{n+2} \xrightarrow{\text{2-D min}} \tau_{n+2}\phi_{n+2}.$$

By not involving τ_{n+1}, Bi-CGSTAB(2) will not suffer the breakdown mentioned above. However, note that it is still prone to breakdown in the BCG ϕ_{n+1} part.

We now show how to overcome this breakdown problem. Recall that CS-CGSTAB already cures the BCG pivot breakdown in step $n+1$, so all we need to do now is to show how to modify CS-CGSTAB so that it will overcome the τ_{n+1} breakdowns as well. The idea is to note that CS-CGSTAB has access to the $A\tau_n\phi_{n+1}r_0$ vector through the relationship $\xi_{n+1} = \sigma_n\phi_{n+1}$. We use it to compute the auxiliary residual $s_{n+2} = \tau_n\phi_{n+2}r_0$ to yield

$$\tau_n\phi_n \xrightarrow{A\tau_n\xi_{n+1}} \tau_n\phi_{n+2} \xrightarrow{\text{2-D min}} \tau_{n+2}\phi_{n+2}.$$

The quantity s_{n+2} is updated by (17) and the 2-D minimization step can be performed by first writing the residual

$$r_{n+2} = s_{n+2} + R\begin{bmatrix}\gamma_1\\ \gamma_2\end{bmatrix},$$

where $R = [\,As_{n+2} \quad A^2s_{n+2}\,] \equiv [\,t_{n+2} \quad v_{n+2}\,]$, and then minimizing $\|r_{n+2}\|$ by solving the system

$$R^TR\begin{bmatrix}\gamma_1\\ \gamma_2\end{bmatrix} = -R^Ts_{n+2}. \tag{34}$$

In the case where A is skew symmetric, the attempt to divide by $\gamma_2 = \omega_{n+1}\omega_{n+2}$ in the 2×2 step can now be performed without breakdown. In particular, if $A = -A^T$, then $\gamma_2 \equiv -(v_{n+2}, s_{n+2})/(v_{n+2}, v_{n+2}) \neq 0$ because the numerator $v_{n+2}^Ts_{n+2} = s_{n+2}^TA^2s_{n+2} > 0$.

We can now form a variant of the CS-CGSTAB algorithm which follows the former method in allowing 1×1 and 2×2 steps and differs only in the 2×2 step, performing instead the minimization over the two degrees of freedom described above. We use the same stepping strategy and approximation scheme to estimate $\|v_{n+2}\|$. We incorporate this into the new variant CS-CGSTAB2. This is a more stable implementation which will not breakdown when A is skew symmetric. Rather, in this case, provided there are no pivot breakdowns, it will always take 2×2 steps and will be equivalent to the Bi-CGSTAB(2) method.

In fact, when only 2×2 steps are taken, the methods Bi-CGSTAB, Bi-CGSTAB(2), CS-CGSTAB, and CS-CGSTAB2 are all mathematically equivalent. In general, the composite step methods differ because they are variable step methods.

Note that in CS-CGSTAB2, the composite step is used to skip over breakdowns in the τ_{n+1} polynomial as well as ϕ_{n+1}. However, if there was no ϕ_{n+1} breakdown, and we took a 2×2 step in CS-CGSTAB2, then it is still possible that there could be pivot breakdown due to ϕ_{n+2}. In principle, we can solve this by applying the composite step idea to Bi-CGSTAB2 and taking a 3×3 step when we foresee possible pivot breakdown because ϕ_{n+3} exists under our assumption of $\det(H^{(0)}) \neq 0$. However, we will not pursue this in this paper.

5. Numerical experiments. All experiments are run in MATLAB 4.0 on a SUN SPARCstation with machine precision of about 10^{-16}. In most cases, as expected, composite step methods behave similarly to their noncomposite step counterparts. In terms of the number of iterations they take to converge, composite step methods are never worse in almost all cases and, in terms of the number of matrix–vector products performed, the cost is minimal. Here, we present a few selected examples where composite step does make a significant improvement.

Example 1. We begin the numerical experiments with a contrived example to illustrate the superior numerical stability of composite step methods over those without composite steps. Let A be a modification of an example found in [23]:

$$A = \begin{pmatrix} M & & \\ & \ddots & \\ & & M\end{pmatrix} \in R^{N\times N}, \quad \text{where } M = \begin{pmatrix}\epsilon & 1\\ -1 & 2\end{pmatrix};$$

TABLE 3
Example 1.

Rel. error in the solution after two steps ($N = 40$)				
ϵ	Bi-CGSTAB	Bi-CGSTAB2	Bi-CGSTAB(2)	CGS
10^{-4}	1.5×10^{-12}	8.6×10^{-13}	2.5×10^{-16}	3.6×10^{-8}
10^{-8}	4.9×10^{-9}	4.7×10^{-9}	1.0×10^{-9}	2.2×10^{0}
10^{-12}	3.0×10^{-5}	7.3×10^{-4}	2.5×10^{-4}	3.0×10^{8}
CS-CGSTAB, CS-CGSTAB2, and CSCGS all converge with errors $\leq 10^{-16}$.				

TABLE 4
Example 2.

Rel. error in the solution after two steps ($N = 40$)					
ϵ	Bi-CGSTAB	Bi-CGSTAB2	Bi-CGSTAB(2)	CGS	CS-CGSTAB
10^{-4}	2.1×10^{-12}	2.9×10^{-13}	2.9×10^{-13}	2.5×10^{-8}	2.3×10^{-13}
10^{-8}	2.0×10^{-8}	bd	1.0×10^{-8}	1.0×10^{0}	7.6×10^{-10}
10^{-12}	1.2×10^{-4}	2.7×10^{1}	1.0×10^{-12}	1.3×10^{8}	bd
bd: Encountered breakdown					
CS-CGSTAB2 converged each time with error $\leq 10^{-16}$.					

i.e., A is an $N \times N$ block diagonal with 2×2 blocks, and $N = 40$. By choosing $b = (1 \;\; 0 \;\; 1 \;\; 0 \;\; \ldots)^T$ and a zero initial guess, we set $\sigma_0 = \epsilon$ and, thus, we can foresee numerical problems with BCG polynomial-based methods such as Bi-CGSTAB, Bi-CGSTAB2, and CGS when ϵ is small. Although these methods converge in two steps in exact arithmetic when $\epsilon \neq 0$, in finite precision, convergence gets increasingly unstable as ϵ decreases. Table 3 shows the relative error in the solution after two steps of BCG, CGS, and Bi-CGSTAB. Note that the loss of significant digits in BCG and Bi-CGSTAB is approximately proportional to $O(\epsilon^{-1})$ and the loss of digits in CGS is proportional to $O(\epsilon^{-2})$. The accuracy of CS-CGSTAB, CS-CGSTAB2, and CSCGS, the composite step CGS algorithm [9], is insensitive to ϵ and these three methods all converge in two steps with errors less than or equal to 10^{-16}.

Example 2. Next we alter Example 1 slightly to show the advantage of CS-CGSTAB2 over CS-CGSTAB. Recall that CS-CGSTAB2 was developed to overcome breakdowns in cases where A is (nearly) skew symmetric. Hence, if we change the last example so that

$$M = \begin{pmatrix} \epsilon & 1 \\ -1 & \epsilon \end{pmatrix},$$

we see that as ϵ gets small, CS-CGSTAB exhibits poor numerical results, whereas CS-CGSTAB2 converges in the first 2×2 step as in Example 1 (see Table 4).

To emphasize the point made in this example, we pick a random skew-symmetric matrix with dimension $N = 20$ and a random right-hand side. All the methods mentioned above either diverge or break down except for CS-CGSTAB2 and BiCGSTAB(2), which achieve residual tolerance 10^{-11} in 24 iterations.

Example 3. We now show an example using a matrix which comes from the Harwell–Boeing set of sparse test matrices [11]. It is a discretization of the convection-diffusion equation:

$$L(u) = -\Delta u + 100(xu_x + yu_y) - 100u$$

on the unit square for a 63×63 grid. We use a random right-hand side, zero initial guess, and no preconditioning. Figure 1 plots the true residual norm for Bi-CGSTAB, Bi-CGSTAB(2), CS-CGSTAB, and CS-CGSTAB2 versus the number of iterations taken, and Figure 2 shows

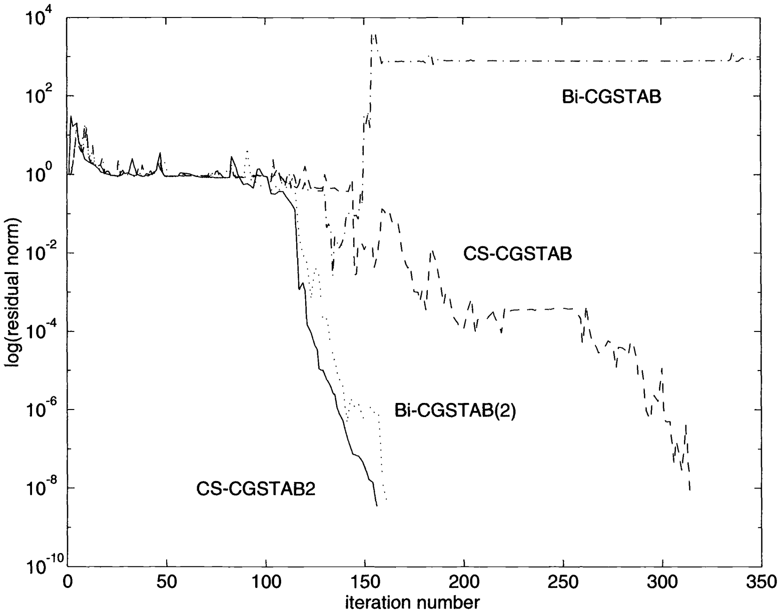


FIG. 1. *Example* 3.

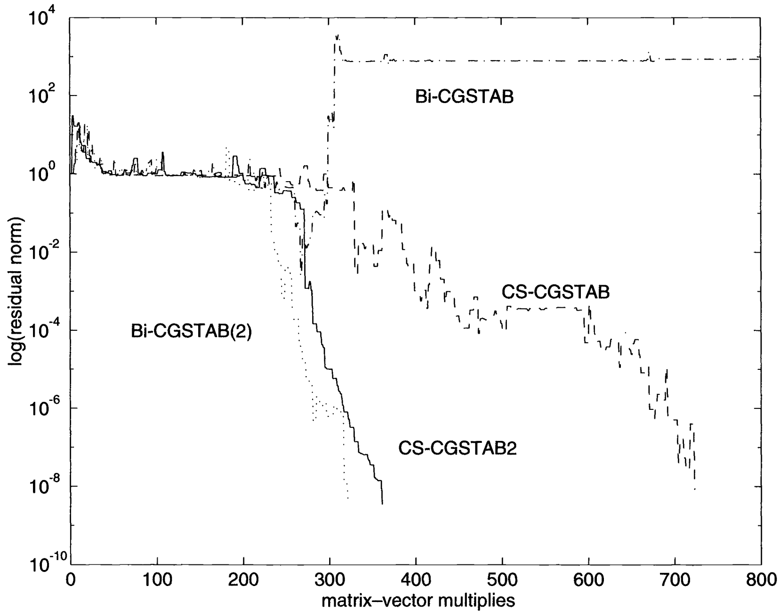


FIG. 2. *Example* 3.

the number of matrix–vector products. The random right-hand side that was used yields some numerical instability for Bi-CGSTAB due to a near pivot breakdown around step 135 which results in convergence stagnation. Taking composite steps in this case overcomes this problem. We also see the advantage of the 2-D minimization steps in the convergence behavior of Bi-CGSTAB(2) and CS-CGSTAB2.

The stepping strategy that we have described and implemented is conservative in that a 2 × 2 step is chosen whenever there is a peak in the residual convergence. If the result of the composite step is only a slight improvement, the extra cost it takes to perform a 2 × 2 step would be wasted. However, in practice, this increase in cost is relatively small. For example, in this particular problem, 96 of the steps taken in CS-CGSTAB are 2 × 2 steps and 50 2 × 2 steps are taken in CS-CGSTAB2. We see that the composite step methods require only about 15% more matrix–vector products in this example.

6. Best approximation results. Until recently there has been very little theory known on the convergence of the BCG algorithm or other related methods. When Bank and Chan introduced CSBCG in [3], they also included a proof of a "best approximation" result for BCG. It is based on an analysis by Aziz and Babuška [1] and is similar to the analysis of the Petrov–Galerkin methods in finite element theory. Specifically, if we let M_k be any symmetric positive definite matrix and define the norm $|||v|||_*^2 \equiv v^t M_k v$, then Bank and Chan showed that the BCG error term $e_k^{BCG} = x - x_k^{BCG} = \phi_k(A)e_0$ can be bounded as follows:

$$|||e_k^{BCG}|||_* \leq (1+\Gamma/\delta) \inf_{\phi_k:\phi_k(0)=1} \|\phi_k(M_k^{\frac{1}{2}}\, A M_k^{-\frac{1}{2}})\|_2 |||e_0|||_*, \tag{35}$$

where Γ and δ are constants independent of k determined from inequalities involving $v \in \mathcal{V}_k$ and $w \in \mathcal{W}_k$, where $\mathcal{V}_k$ and $\mathcal{W}_k$ are the Krylov subspaces generated by the Lanczos method at the kth step:

$$|w^T A v| \leq \Gamma |||v|||_* |||w|||_*,$$
$$\inf_{\substack{v\in\mathcal{V}\| \\ |||v|||_*=1}} \; \sup_{\substack{w\in\mathcal{W}\| \\ |||w|||_*=1}} w^T A v \geq \delta_k > \delta > 0.$$

Moreover, if we define the Lanczos tridiagonal matrix $T_k = W_k^T A V_k$ and its LU-factorization $T_k = L_k D_k U_k$, and define $M_k \equiv W_k U_k^T (D_k^T D_k)^{\frac{1}{2}} U_k W_k^T$, then $\Gamma = \delta = 1$, also shown in [3].

This result establishes convergence of BCG when there are no breakdowns, because, in this case, M_k is well defined and symmetric positive definite. In the case where there are breakdowns, the tridiagonal matrix T_k would be singular and the resulting M_k would not be positive definite. However, the convergence result can be extended to cover these situations using methods such as composite step to handle breakdown. Specifically, assuming no Lanczos breakdowns, the composite step approach yields a valid M_k matrix based on a factorization of T_k which may involve 2 × 2 block and, hence, the above result applies to the error e_k corresponding to the well-defined iterates x_k [2]. In principle, if we add a look-ahead method to handle the Lanczos breakdowns to this, we can prove convergence of BCG for cases where both breakdowns occur.

Note that, in general, simple upper bounds for the term

$$\inf_{\phi_k:\phi_k(0)=1} \|\phi_k(M_k^{\frac{1}{2}}\, A M_k^{-\frac{1}{2}})\|_2 \tag{36}$$

are known only for special cases. For example, if we assume that the eigenvalues of A are contained in an ellipse in the complex plane which does not contain the origin, then, due to a result by Manteuffel [22], the quantity (36) can be bounded by a value dependent on the foci of the ellipse.

The product methods discussed earlier (CGS and Bi-CGSTAB) both involve the BCG polynomial. Hence, we can use the result in (35) to establish bounds on these methods as well. For the following proofs, we define $\tilde{A}_k \equiv M^{\frac{1}{2}}\, A M_k^{-\frac{1}{2}}$. We first prove a lemma which will be used in the derivation of both bounds for CGS and Bi-CGSTAB.

LEMMA 1. *For any matrix $A \in R^{N\times N}$, vector $v \in R^N$, and polynomial p,*

$$|||p(A)v|||_* \le \|p(\tilde{A}_k)\|_2|||v|||_*.$$

Proof. By definition, $|||w|||_* = \|M_k^{\frac{1}{2}} w\|_2$ and, thus, $|||p(A)v|||_* = \|M_k^{\frac{1}{2}} p(A)v\|_2 = \|M_k^{\frac{1}{2}} p(A)M_k^{-\frac{1}{2}} M_k^{\frac{1}{2}} v\|_2 \le \|p(M_k^{\frac{1}{2}} AM_k^{-\frac{1}{2}})\|_2\|M_k^{\frac{1}{2}} v\|_2 = \|p(\tilde{A}_k)\|_2|||v|||_*$. □

We now use this result to estimate a bound on the CGS method and show the squaring effect on the convergence rate.

THEOREM 1. *Let $e_k^{CGS} = \phi_k^2(A)e_0$. Then*

$$|||e_k^{CGS}|||_* \le c_1 \left(\inf_{\phi_k:\phi_k(0)=1} \|\phi_k(\tilde{A}_k)\|_2 \right)^2 |||e_0|||_*.$$

Proof. Applying Lemma 1 to $|||e_k^{CGS}|||_*$, we get

$$\begin{aligned}|||e_k^{CGS}|||_* &= |||\phi_k^2(A)e_0|||_* \le \|\phi_k(\tilde{A}_k)\|_2|||\phi_k(A)e_0|||_* \\ &= \|\phi_k(\tilde{A}_k)\|_2|||e_k^{BCG}|||_* \le c_1 \left(\inf_{\phi_k:\phi_k(0)=1} \|\phi_k(\tilde{A}_k)\|_2 \right)^2 |||e_0|||_*. \qquad \square\end{aligned}$$

Next we show convergence of the Bi-CGSTAB method and how the convergence rate of BCG can be improved by the effect of the steepest descent.

THEOREM 2. *Let $e_k^{Bi\text{-}CGSTAB} = \tau_k(A)\phi_k(A)e_0$, and define S to be the symmetric part of $\tilde{A}_k$ (i.e., $S = \frac{1}{2}(\tilde{A}_k + \tilde{A}_k^T)$). Then if S is positive definite,*

$$|||e_k^{Bi\text{-}CGSTAB}|||_* \le c_2 \left(1 - \frac{\lambda_{\min}(S)^2}{\lambda_{\max}(\tilde{A}_k^T\tilde{A}_k)} \right)^{\frac{k}{2}} \left(\inf_{\phi_k:\phi_k(0)=1} \|\phi_k(\tilde{A}_k)\|_2 \right) |||e_0|||_*.$$

Proof. First note that we can bound

$$\min_{\tau_k\in P_k} \|\tau_k(\tilde{A}_k)\|_2 \le \left(1 - \frac{\lambda_{\min}(S)^2}{\lambda_{\max}(\tilde{A}_k^T\tilde{A}_k)} \right)^{\frac{k}{2}}$$

by applying the proof in Theorem 3.3 of Eisenstat, Elman, and Schultz [12] to the matrix $\tilde{A}_k$. Combining this fact with Lemma 1, we can establish the following bound:

$$\begin{aligned}|||e_k^{Bi\text{-}CGSTAB}|||_* &= \min_{\tau_k\in P_k} |||\tau_k(A)\phi_k(A)e_0|||_* \le \min_{\tau_k\in P_k} \left(\|\tau_k(M_k^{\frac{1}{2}} AM_k^{-\frac{1}{2}})\|_2|||\phi_k(A)e_0|||_* \right) \\ &\le \left(\min_{\tau_k\in P_k} \|\tau_k(\tilde{A}_k)\|_2 \right) |||\phi_k(A)e_0|||_* \le \left(1 - \frac{\lambda_{\min}(S)^2}{\lambda_{\max}(\tilde{A}_k^T\tilde{A}_k)} \right)^{\frac{k}{2}} |||e_k^{BCG}|||_* \\ &\le c_2 \left(1 - \frac{\lambda_{\min}(S)^2}{\lambda_{\max}(\tilde{A}_k^T\tilde{A}_k)} \right)^{\frac{k}{2}} \left(\inf_{\phi_k:\phi_k(0)=1} \|\phi_k(\tilde{A}_k)\|_2 \right) |||e_0|||_*. \qquad \square\end{aligned}$$

Recently, Barth and Manteuffel [4] have shown that the constant $(1 + \Gamma/\delta)$ in (35) can be improved to one. In other words, e_k^{BCG} is minimized over $K_n(r_0)$ in the $|||\cdot|||_*$ norm. Correspondingly, the constants c_1 and c_2 in Theorems 1 and 2 can be improved to one.

A. Appendix. Here, we present the proof of the nonsingularity of the coefficient matrix in (19) and (20).

LEMMA A.1. *Assume that the Lanczos process underlying BCG does not break down; i.e.,* $\rho_i \neq 0$, $i = 0, 1, \ldots, n$, *and* $\tilde{z}_{n+1}^T z_{n+1} \neq 0$. *If* $\sigma_n^{BCG} = 0$, *then the coefficient matrix in* (19) *and* (20) *is nonsingular.*

Proof. Since $\sigma_n^{Bi\text{-}CGSTAB} = \frac{1}{\mu_n}\, \sigma_n^{BCG}$, where μ_n is as defined in §3, it suffices to show that if $\sigma_n^{Bi\text{-}CGSTAB} = 0$, then the 2×2 matrix in (19) and (20) is nonsingular. Note that $\sigma_n^{Bi\text{-}CGSTAB} = \tilde{r}_0^T A p_n^{Bi\text{-}CGSTAB} = (\tilde{r}_0, \vartheta \tau_n \psi_n r_0)$, the (1,1) element of the 2×2 matrix in question. Note also the relationship of the off diagonal elements in the case where $\sigma_n^{BCG} = 0:$ $(\tilde{r}_0, \vartheta \tau_n \xi_{n+1} r_0) = (\tilde{r}_0, \vartheta \tau_n (\sigma_n^{BCG} \phi_n - \rho_n \vartheta \psi_n) r_0) = -\rho_n (\tilde{r}_0, \vartheta^2 \tau_n \psi_n r_0)$.

Since $\rho_n \neq 0$, nonsingularity follows from showing that $(\tilde{r}_0, \vartheta^2 \tau_n \psi_n r_0) \neq 0$ when $\sigma_n^{BCG} = 0$. Analogous to the derivation of (26) and (27), $(\tilde{r}_0, A^2 \tau_n(A) \psi_n(A) r_0) = \frac{1}{\mu_n}\, (\psi_n(A^T)\tilde{r}_0, A^2 \psi_n(A) r_0)$. Using $0 \neq \tilde{z}_{n+1}^T z_{n+1} \equiv \rho_n^2 (\psi_n \tilde{r}_0, A^2 \psi_n r_0)$ and $\rho_n \neq 0$, we now have $(\psi_n \tilde{r}_0, A^2 \psi_n r_0) \neq 0$, thus completing our proof. □

REFERENCES

[1] A. K. AZIZ AND I. BABUŠKA, *Part I, Survey lectures on the mathematical foundations of the finite element method,* in The Mathematical Foundations of the Finite Element Method with Applications to Partial Differential Equations, Academic Press, New York, 1972, pp. 1–362.

[2] R. E. BANK AND T. F. CHAN, *An analysis of the composite step bi-conjugate gradient algorithm for solving nonsymmetric systems*, Numer. Math., 66(1993), pp. 295–319.

[3] ———, *A composite step bi-conjugate gradient algorithm for nonsymmetric linear systems*, Numer. Algorithms, 7(1994), pp. 1–16.

[4] T. BARTH AND T. MANTEUFFEL, *Variable metric conjugate gradient methods*, in Proceedings of the Tenth International Symposium on Matrix Analysis and Parallel Computing, Keio University, Yokohama, Japan, March, 1994.

[5] C. BREZINSKI AND H. SADOK, *Lanczos-type algorithms for solving systems of linear equations*, Appl. Numer. Math., 11(1993), pp. 443–473.

[6] C. BREZINSKI AND M. REDIVO-ZAGLIA, *Breakdowns in the computation of orthogonal polynomials*, in Nonlinear Numerical Methods and Rational Approximation, A. Cuyt, ed., Kluwer, Dordrecht, the Netherlands, 1994, pp. 49–59.

[7] C. BREZINSKI, M. REDIVO-ZAGLIA, AND H. SADOK, *A breakdown-free Lanczos type algorithm for solving linear systems*, Numer. Math., 63(1992), pp. 29–38.

[8] ———, *Avoiding breakdown and near-breakdown in Lanczos type algorithms*, Numer. Algorithms, 1(1991), pp. 261–284.

[9] T. F. CHAN AND T. SZETO, *A composite step conjugate gradients squared algorithm for solving nonsymmetric linear systems*, Numer. Algorithms, 7(1994), pp. 17–32.

[10] A. DRAUX, *Polynômes orthogonaux formels*, Lecture Notes in Math., 974, Springer-Verlag, Berlin, 1983.

[11] I. S. DUFF, R. G. GRIMES, AND J. G. LEWIS, *Sparse matrix test problems*, ACM Trans. Math. Software, 15(1989), pp. 1–14.

[12] S. C. EISENSTAT, H. C. ELMAN, AND M. H. SCHULTZ, *Variational iterative methods for nonsymmetric systems of linear equations*, SIAM J. Numer. Anal., 20(1983), pp. 345–357.

[13] R. FLETCHER, *Conjugate gradient methods for indefinite systems*, in Numerical Analysis, Lecture Notes in Math., 506, G. A. Watson, ed., Springer-Verlag, Berlin, 1976, pp. 73–89.

[14] R. W. FREUND AND N. M. NACHTIGAL, *QMR: A quasi-minimal residual method for non-Hermitian linear systems*, Numer. Math., 60(1991), pp. 315–339.

[15] ———, *A Look Ahead TFQMR Method*, Presented at the Cornelius Lanczos International Centenary Conference, Raleigh, NC, December 1993.

[16] R. W. FREUND, M. H. GUTKNECHT, AND N. M. NACHTIGAL, *An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices*, SIAM J. Sci. Comput., 13 (1992), pp. 137–158.

[17] M. H. GUTKNECHT, *The unsymmetric Lanczos algorithms and their relations to Pade approximation, continued fraction and the QD algorithm*, in Proc. Copper Mountain Conference on Iterative Methods, Book 2, Breckenridge, CO, 1990.

[18] ———, *Variants of BiCGSTAB for matrices with complex spectrum*, SIAM J. Sci. Comput., 14 (1993), pp. 1020–1033.

[19] M. R. HESTENES AND E. STIEFEL, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Stand., 49(1952), pp. 409–436.

[20] W. JOUBERT, *Generalized Conjugate Gradient and Lanczos Methods for the Solution of Nonsymmetric Systems of Linear Equations*, Ph.D. thesis, The University of Texas at Austin, Austin, TX, 1990.

[21] C. LANCZOS, *Solution of linear equations by minimized iterations*, J. Res. Nat. Bur. Stand., 49 (1952), pp. 33–53.

[22] T. MANTEUFFEL, *An Iterative Method for Solving Nonsymmetric Linear Systems with Dynamic Estimation of Parameters*, Ph.D. thesis, University of Illinois, Urbana, IL, 1975.

[23] N. M. NACHTIGAL, S. C. REDDY, and L. N. TREFETHEN, *How Fast Are Nonsymmetric Matrix Iterations?*, Tech. report, Department of Mathematics, MIT, Cambridge, MA, 1990.

[24] B. N. PARLETT, D. R. TAYLOR, AND Z. A. LIU, *A look-ahead Lanczos algorithm for unsymmetric matrices*, Math. Comput., 44(1985), pp. 105–124.

[25] G. SLEIJPEN AND D. FOKKEMA, *BICGSTAB(L) for linear equations involving unsymmetric matrices with complex spectrum*, ETNA, 1(1993), pp. 11–32.

[26] P. SONNEVELD, *CGS, a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 10(1989), pp. 36–52.

[27] C. H. TONG, *A Comparative Study of Preconditioned Lanczos Methods for Nonsymmetric Linear Systems*, Tech. report SAND91-8402 UC-404, Sandia National Laboratories, Albuquerque, NM, 1992.

[28] H. A. VAN DER VORST, *Bi-CGSTAB: A fast and smoothly converging variant of bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 13(1992), pp. 631–644.

[29] ———, *The convergence behaviour of preconditioned CG and CG-S in the presence of rounding errors*, in Preconditioned Conjugate Gradient Methods, O. Axelsson and L. Yu. Kolotilina, eds., Lecture Notes in Math., 1457, Springer-Verlag, Berlin, 1990.